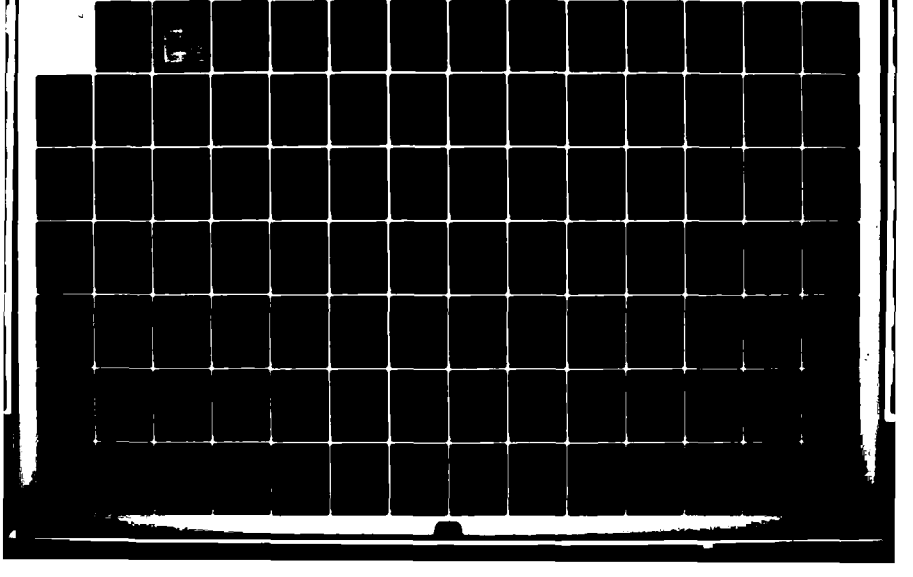
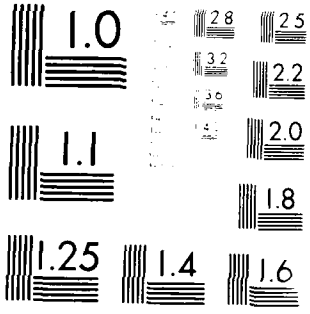
AD-A142 394 A NUMERICAL ALGORITHM FOR CHAINED AGGREGATION AND MODIFIED CHAINED AGGREGATION(U) ILLINOIS UNIV AT URBANA DECISION AND CONTROL LAB H S THARP SEP 83 DC-62

UNCLASSIFIED N00014-79-C-0424 F/G 12/1 NL

1/2





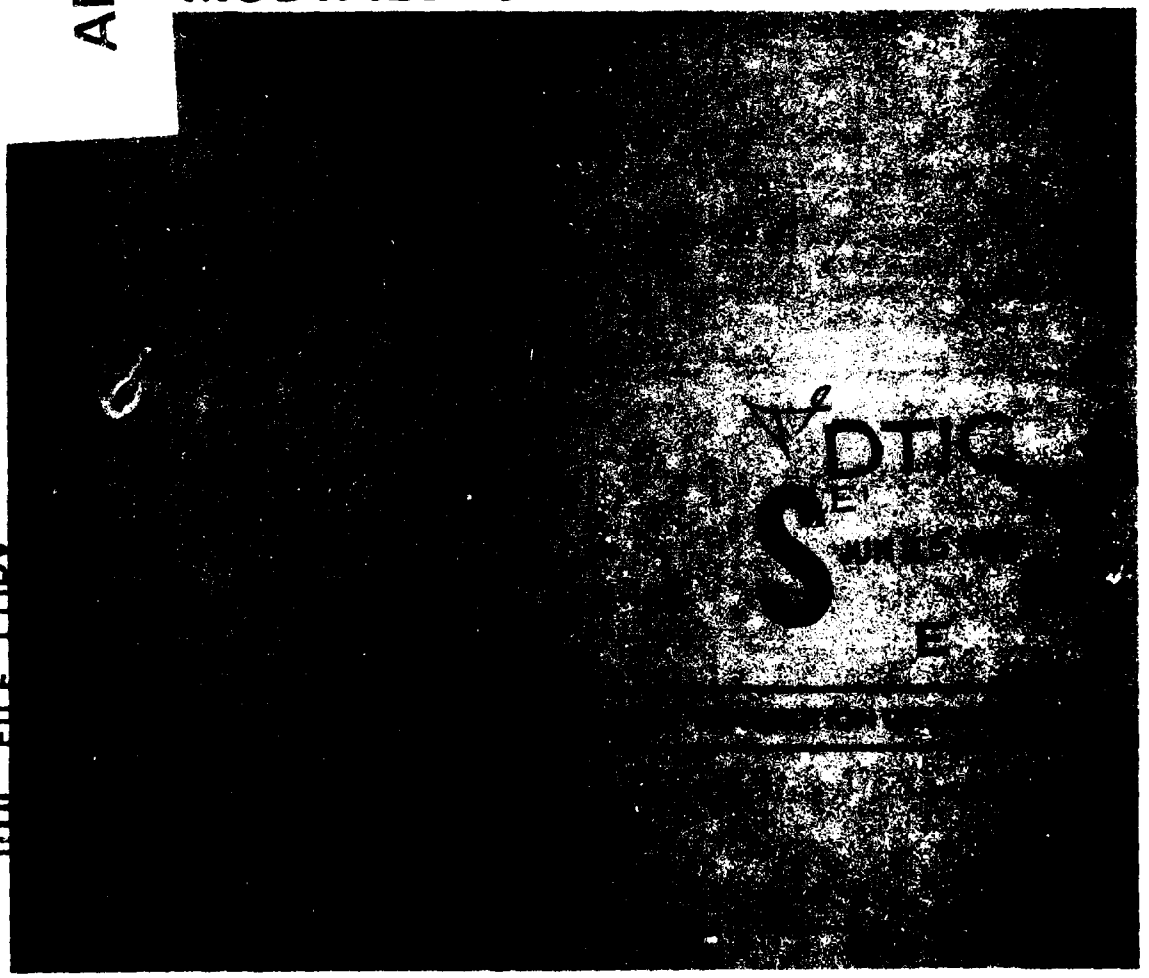
(12)

REPORT DC-62 SEPTEMBER 1983

COORDINATED SCIENCE LABORATORY

DECISION AND CONTROL LABORATORY

A NUMERICAL ALGORITHM FOR CHAINED AGGREGATION AND MODIFIED CHAINED AGGREGATION



UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A142 394	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A NUMERICAL ALGORITHM FOR CHAINED AGGREGATION AND MODIFIED CHAINED AGGREGATION		5. TYPE OF REPORT & PERIOD COVERED Technical Report 6. PERFORMING ORG. REPORT NUMBER R-996(DC-62);UILU-ENG-83-2217
7. AUTHOR(s) Hal Stanley Tharp		8. CONTRACT OR GRANT NUMBER(s) N00014-79-C-0424; NSF ECS 82-17631
9. PERFORMING ORGANIZATION NAME AND ADDRESS Coordinated Science Laboratory University of Illinois at Urbana-Champaign Urbana, Illinois 61801		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Joint Services Electronics Program National Science Foundation		12. REPORT DATE September 1983 13. NUMBER OF PAGES 86
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Chained aggregation
Generalized Hessenberg Representation
Modified chained aggregation
Orthogonal matrices
Singular value decomposition
Large space structure
Output feedback
Reduced-order model

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A computer implementation of chained aggregation and modified chained aggregation using orthogonal transformations is presented. The numerical advantages associated with orthogonal matrices are highlighted. The developed algorithm is then employed to identify a reduced-order model of a large space structure. With the reduced-order model, an output feedback design is carried out. The design is shown to produce an acceptable result on the full order model.

DD FORM 1 JAN 73 1473

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

A NUMERICAL ALGORITHM FOR CHAINED AGGREGATION AND MODIFIED CHAINED AGGREGATION

BY

HAL STANLEY THARP

B.S., University of Missouri-Rolla, 1981

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1983

Thesis Advisor: Professor William R. Perkins

Urbana, Illinois

ACKNOWLEDGEMENTS

The author would like to thank his advisor Professor W. R. Perkins for the many helpful discussions and suggestions during the course of this work. Thanks are also extended to Professor D. K. Lindner whose insight and comments provided the stimulus to complete the development of the algorithm.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The chained aggregation procedure was first introduced in [1] in the context of reduced-order modelling of large-scale systems. The chained aggregation procedure transforms the original system into the Generalized Hessenberg Representation (GHR). Once placed in the GHR, the reduced-order modelling analysis is carried out.

Since the introduction of the chained aggregation procedure, much more research has been done [2,3,4,5]. The work that has followed has not been constrained to the reduced-order modelling problem. But, it has expanded the possible applications of the chained aggregation procedure to include many control system problems. For a thorough discussion of chained aggregation and the GHR for control system design, the author recommends consulting [6], where this issue has been specifically addressed in a geometric setting.

The algorithm developed in this thesis uses the numerical advantages associated with orthogonal matrices to implement the basic chained aggregation procedure. Included in the algorithm is an enhanced procedure, called modified chained aggregation (MCA), which has been introduced in earlier literature [2,6,7]. Several numerical advantages of orthogonal matrices are included herein; more discussion may be found in [8,9,10,11,12].

The orthogonal matrices are obtained during the algorithm by using the singular value decomposition (SVD) of particular matrices. Briefly, the SVD is used to identify the linearly independent and linearly dependent rows and columns of the particular matrix. With the identification, the decomposition generates orthonormal vectors spanning the subspaces associated with the above sets of rows and columns. These orthonormal vectors may then be judiciously grouped to form orthogonal matrices to be used within the algorithm.

The organization of this thesis is as follows. Chapter 2 presents the SVD and steps through the chained aggregation algorithm showing how SVD has been incorporated. The development of the modified chained aggregation algorithm is contained in Chapter 3. The chained aggregation algorithm is used in Chapter 4 to help in the design of an output feedback controller for a particular large flexible space structure. Chapter 5 presents the conclusion.

A summary of definitions and the notation used throughout this thesis follow. Given a subspace $\mathbf{S}$, denoted by bold, capital roman letters, its orthogonal complement will be represented by $\mathbf{S}^{\perp}$. The range space and the null space of a matrix will be denoted by $\mathbf{R}[\cdot]$ and $\mathbf{N}[\cdot]$, respectively. $A^{\dagger}$ is used to represent the pseudo-inverse of the matrix A. The transpose of a matrix A will be denoted by A^{T}. The set of all mxn matrices of rank r with coefficients in the real number field, R, will be denoted by R_r^{mxn}. The working precision of the computer will be represented by ε. On a given computer, the value of ε equals the smallest number which when added to one equals one. The singular values of a matrix will be denoted by σ_i. The matrix norms $\|\cdot\|_2$ and $\|\cdot\|_F$ correspond to the matrix 2-norm and the

matrix Frobenius norm, respectively. If $A \in R^{m \times n}$, then

$$||A||_2 = \underset{||x||_2=||y||_2=1}{\text{maximize}} |y^T A x| ,$$

where

$$||x||_2 = \left(\sum_{i=1}^{n} \xi_i^2 \right)^{1/2} = (x^T x)^{1/2}$$

$$x = (\xi_1, \xi_2, \ldots, \xi_n)^T .$$

$$||A||_F = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2 \right)^{1/2} .$$

The dimension of a vector space will be denoted by $d(\cdot)$. The notation

$$sp \begin{bmatrix} 0 \\ X \end{bmatrix}$$

represents the vector space spanned by the non-zero elements of the columns of X as they vary over the real numbers. The rank of a matrix will be denoted by $\rho[\cdot]$.

CHAPTER 2

SINGULAR VALUE DECOMPOSITION AND THE CHAINED AGGREGATION ALGORITHM

2.1. Introduction

Chained aggregation as introduced in [1] identifies the information structure of the system by aggregating the system with respect to the output. Once aggregated, the system exhibits the Generalized Hessenberg Representation (GHR) structure which has been shown to stimulate many possible design procedures [1,2,3,4,5,6].

A design procedure used in this thesis is model reduction. Model reduction arises out of the simple interconnecting structure displayed in the GHR and by identifying the feedback coupling between subsystems which is weak or nonexistent. The resulting model retains the strongly observable modes from the outputs rather than retaining the dominant system modes as in modal reduction. Throughout the following discussion the subsystem composed of the strongly observable modes will be referred to as the aggregate subsystem, while the residual subsystem will refer to the remaining subsystem.

Associated with the chained aggregation procedure are many state space transformations. It is shown below these transformations can be performed using orthogonal matrices which are numerically robust, in contrast to the nonorthogonal transformations described in most of the previous literature [1,2,3,4,7].

The following section reviews the singular value decomposition (SVD) along with some of its properties. The SVD is then incorporated into the steps of the chained aggregation algorithm as shown in Section 3. Section 4 highlights a consequence of the chained aggregation algorithm, its ability to check the observability and controllability of a system.

2.2. Singular Value Decomposition and Its Properties

The SVD will be introduced by the following theorem.

Theorem 2.2.1[8]: Let $A \in R_r^{mxn}$. Then there exist orthogonal matrices $U \in R^{mxm}$ and $V \in R^{nxn}$ such that

$$A = U \Sigma V^T \quad , \tag{2.2.1}$$

where

$$\Sigma = \begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix}$$

and $S = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$ with $\sigma_1 \geq \ldots \geq \sigma_r > 0$.

Proof: See [8].

The product $U \Sigma V^T$ is the singular value decomposition of the A matrix. The numbers $\sigma_1, \sigma_2, \ldots, \sigma_r$ together with $\sigma_{r+1} = \ldots = \sigma_n = 0$ are called the singular values of A and are the positive square roots of the eigenvalues of A^TA.

The following well-known properties of SVD make it useful in the chained aggregation algorithm.

Two different matrix norms associated with $\sigma_1,\ldots,\sigma_n$ can easily be defined:

$$||A||_2 = \sigma_1$$

$$||A||_F = (\sigma_1^2 + \ldots + \sigma_n^2)^{1/2} .$$

Our main interest in singular values will be for rank determination. Thus, knowing that the singular values are not very sensitive to perturbations in the matrix insures a good rank determination. The change in the singular values are known to be bounded by the magnitude of the matrix perturbation.

Theorem 2.2.2[11]: Let $A,B \in R^{mxn}$ have singular values $\sigma_1 \geq \ldots \geq \sigma_n$ and $\tau_1 \geq \ldots \geq \tau_n$, respectively. Then

$$|\sigma_i - \tau_i| \leq ||A - B||_2 \qquad (i = 1,2,\ldots,n).$$

Proof: See [11].

The concern over matrix perturbations is related to the fact that infinite precision arithmetic cannot be performed on a computer. For this reason, when the singular values of a matrix A are desired, the computed singular values are actually the exact singular values of a matrix slightly perturbed from A, say A + E. A more extensive discussion can be found in [8,10].

To determine the rank of a matrix, all of its singular values are compared with $\gamma = \varepsilon \cdot ||A||$, where the particular norm used can be selected by

the user; the number of σ_i's greater than γ determines the rank [10]. Looked at in this way, the smallest singular value, σ_r, greater that γ gives a measure as to how far away the A matrix is from another matrix of smaller rank.

Another important outcome of SVD is the identification of the four fundamental subspaces [9] of a matrix. Suppose $F \in R_r^{mxn}$ has an SVD given by

$$F = U \Sigma V^T = \begin{bmatrix} U_1 , U_2 \end{bmatrix} \begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix}, \qquad (2.2.2)$$

where $S = \text{diag}(\sigma_1,\ldots,\sigma_r)$ with $\sigma_1 \geq \ldots \geq \sigma_r > 0$ and U and V are partitioned compatably, i.e., $U_1 \in R^{mxr}$, $V_1 \in R^{nxr}$, etc. With this notation U_1, U_2 and V_1, V_2 produce orthonormal bases for the four fundamental subspaces, $\mathcal{R}[F]$, $\mathcal{R}^{\perp}[F]$, $\mathcal{N}^{\perp}[F]$, $\mathcal{N}[F]$ [8]. Figure 2.1 redrawn from [8] relates these subspaces.

Associated with the matrix $F \in R^{mxn}$ are two different vector spaces, R^m and R^n. Figure 2.1 illustrates how the F matrix can induce direct sum decompositions of these two vector spaces. In R^n the columns of V_2 form an orthogonal basis for the $\mathcal{N}[F]$ while the columns of V_1 span the remainder of R^n with an orthogonal basis. Similarly, the columns of U_1 form an orthogonal basis for the $\mathcal{R}[F]$ in R^m while the columns of U_2 span the remainder of R^m with an orthogonal basis. The mappings between these subspaces is also indicated. The F matrix maps the space spanned by V_1 into the $\mathcal{R}[F]$, i.e., the space spanned by U_1. The pseudo-inverse of F performs a map in the opposite direction. The space spanned by the columns of V_2 is mapped into the origin in R^m by the F matrix. The space spanned by the

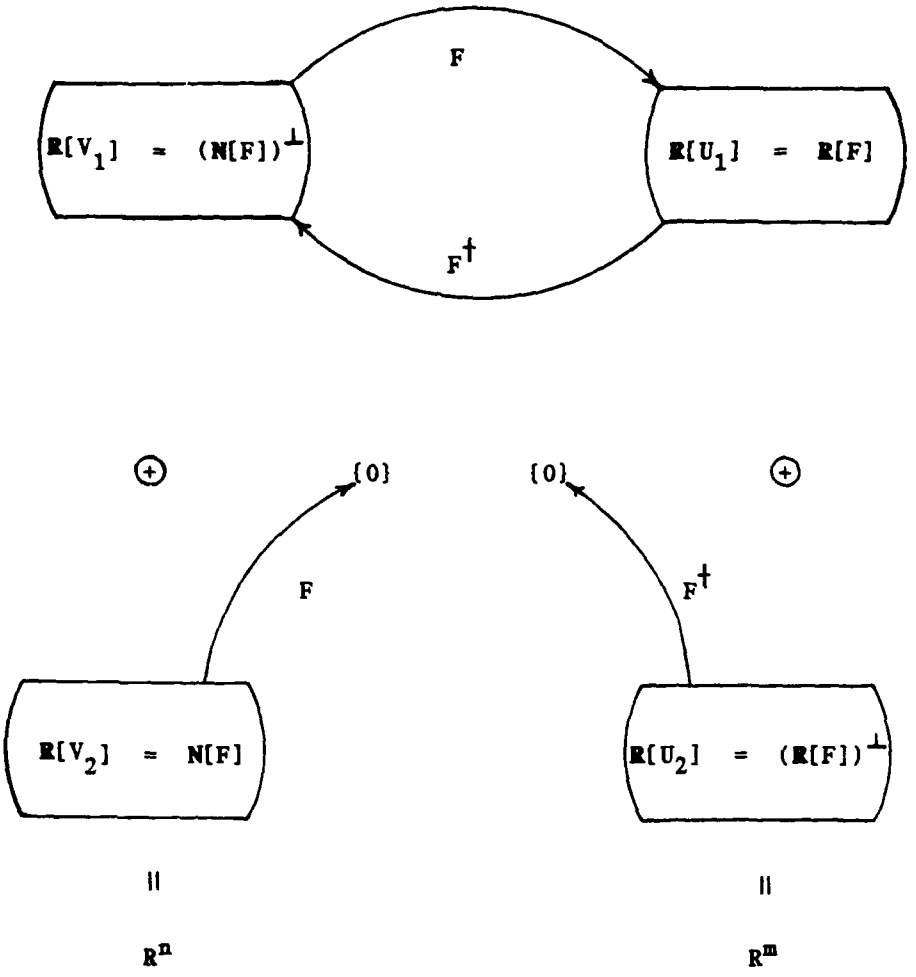


Figure 2.1: The Four Fundamental Subspaces

columns of U_2 is mapped into the origin in R^n by $F^\dagger$.

The pseudo-inverse of F can be obtained from the SVD of F

$$F^\dagger = V \Sigma^\dagger U^T = \begin{bmatrix} V_1 , V_2 \end{bmatrix} \begin{bmatrix} S^{-1} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} U_1^T \\ U_2^T \end{bmatrix}, \qquad (2.2.3)$$

where $S^{-1} = \text{diag}(\frac{1}{\sigma_1},\ldots,\frac{1}{\sigma_r})$, $\sigma_1 \geq \ldots \geq \sigma_r > 0$, and U and V are obtained from (2.2.2).

In general, SVD is the only numerically reliable method of generating basis for these subspaces, since U and V are partitioned into $[U_1 , U_2]$ and $[V_1 , V_2]$, according to the smallest singular value [8].

Further discussion of Figure 2.1 will be postponed until after the chained aggregation algorithm has been presented.

The advantage of using SVD within the chained aggregation algorithm lies in the fact that the bases vectors generated by SVD are orthonormal. Thus, constructing transformation matrices from these orthonormal vectors results in orthogonal transformations. Several well-known numerical advantages associated with orthogonal matrices useful in the chained aggregation algorithm are:

(1) Orthogonal matrices are easy to invert, $U^{-1} = U^T$.

(2) Orthogonal matrices are perfectly conditioned with respect to the 2-norm, $||AU||_2 = ||A||_2$.

(3) Orthogonal matrices lend themselves to backward error analyses [11]. For example, suppose an error F is introduced into the result of an

orthogonal transformation. Let $E = UFU^T$. From the second characteristic above

$$||E||_2 = ||UFU^T||_2 = ||FU^T||_2 = ||F||_2$$

and

$$U^T(A + E)U = U^TAU + U^TEU = U^TAU + F .$$

In other words, a perturbation in the result can be accounted for by a perturbation of the same magnitude in the original problem. This guarantees that if there exists an uncertainty in the original A matrix of magnitude δ, then the resulting transformed system will have an uncertainty of the same δ magnitude.

(4) Orthogonal transformations only rotate the axes, each axis maintaining its exact relation to the others throughout [12]. It should be mentioned that not all orthogonal matrices represent pure rotations as can be seen from a simple example. Consider the orthogonal matrix

$$P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} .$$

This matrix reflects every point (x,y) into its mirror image (y,x) across the 45° line y = x, which is not a true rotation.

In [1] orthogonal matrices have not been used for the transformation matrices and consequently the matrices used can be very ill conditioned and result in a numerically unstable algorithm.

2.3. The Chained Aggregation Algorithm

The chained aggregation algorithm transforms any given system of the form

$$\dot{x} = Ax + Bu \qquad (2.3.1a)$$

$$y = Cx \ , \qquad (2.3.1b)$$

where $A \in R^{n\times n}$, $B \in R^{n\times m}$, and $C \in R^{r_1 \times n}$ into the GHR using only the information structure of the system. The GHR is given by

$$\dot{z} = Fz + Gu \qquad (2.3.2a)$$

$$y = Dz \ , \qquad (2.3.2b)$$

where

$$F = \begin{bmatrix} F_{11} & F_{12} & 0 & \cdots & 0 & 0 \\ F_{21} & F_{22} & F_{23} & 0 & \cdots & 0 & 0 \\ \vdots & & & & & & \vdots \\ F_{j1} & F_{j2} & \cdots & F_{jj} & F_{j,j+1} & 0 & \cdots & 0 \\ \vdots & & & & & & \vdots \\ F_{k-2,1} & F_{k-2,2} & \cdots & & & & F_{k-2,k-1} & 0 \\ F_{k-1,1} & F_{k-1,2} & \cdots & & & & F_{k-1,k-1} & F_{k-1,k} \\ F_{k,1} & F_{k,2} & \cdots & & & & F_{k,k-1} & F_{k,k} \end{bmatrix}$$

$$G = \begin{bmatrix} G_1 \\ G_2 \\ \vdots \\ G_k \end{bmatrix}, \qquad D = \begin{bmatrix} D_1 & 0 & \cdots & 0 \end{bmatrix}$$

with $F_{i,i} \in R^{r_i x r_i}$, $r_i \geq r_{i+1}$, $\sum_{i=1}^{k} r_i = n$, and $D_1 \in R^{r_1 x r_1}$. The following steps comprise the chained aggregation algorithm.

2.3.1. Algorithm

Let $A \in R^{nxn}$, $B \in R^{nxm}$, and $C \in R^{r_1 xn}$ with C having full row rank.

Initialization: $A^1 = A$, $C = C$, $\tau = 0$, $\rho = n$, $\gamma = \varepsilon \|A_{11}\|$, and $i = 1$.

Step 1: Perform a column compression [10] on the C matrix by using SVD. The linearly independent columns of C will be compressed to the left by post multiplying C by the V matrix resulting from SVD. If $V^i = [V_1^i , V_2^i]$ with $V_1^i \in R^{nxr_1}$ and $V_2^i \in R^{nx(n-r_1)}$, then

$$C = U^i \Sigma^i V^{iT}$$

results in

$$CV^i = C[V_1^i , V_2^i] = [D_1 \quad 0] .$$

Step 2: Initialize the transformation matrix

$$T^i = V^{iT} .$$

Step 3: Perform a state space transformation on the A^i matrix. This transformation can be broken down into four smaller and separate transformations. Since at most only two of the four smaller transformations are needed, this allows for fewer calculations during the execution of the algorithm.

$$V^{iT}A^iV^i = \begin{bmatrix} V_1^{iT}A^iV_1^i & V_1^{iT}A^iV_2^i \\ V_2^{iT}A^iV_1^i & V_2^{iT}A^iV_2^i \end{bmatrix}$$

$$= \begin{bmatrix} A_{11}^i & A_{12}^i \\ A_{21}^i & A_{22}^i \end{bmatrix} .$$

Let $\rho = \rho - r_i$.

Compute only $A_{12}^i = V_1^{iT} A^i V_2^i \in R^{r_i x \rho}$ at this step.

Step 4: Compress the columns of A_{12}^i using SVD.

$$A_{12}^i = U^{i+1} \Sigma^{i+1} V^{(i+1)T}$$

$$A_{12}^i [V_1^{i+1}, V_2^{i+1}] = [F_{i,i+1}, 0] \quad .$$

where $F_{i,i+1} \in R^{r_i x r_{i+1}}$ and $V^{i+1} \in R^{\rho x \rho}$.

Step 5: Check the singular values of A_{12}^i.

If all σ_j's $< \gamma$ $(j = 1,2,\ldots,r_i)$, then exit 1.

If all σ_j's $\geq \gamma$ $(j = 1,2,\ldots,r_{i+1} \leq r_i)$, then exit 2.

Otherwise, continue with $\tau = \tau +$ (number of σ_j's $\geq \gamma$).

Step 6: Update the transformation matrix.

$$T^{i+1} = \begin{bmatrix} I_\tau & 0 \\ 0 & V^{(i+1)T} \end{bmatrix} T^i \quad ,$$

where I_τ implies $I \in R^{\tau x \tau}$.

Step 7: Calculate the A_{22}^i submatrix.

$$A_{22}^i = V_2^{iT} A^i V_2^i \in R^{\rho x \rho} \quad .$$

Step 8: Let $A^{i+1} = A_{22}^i$, $i = i + 1$. Go to Step 3.

Exit 1: The system aggregates. $F = TAT^T$, $G = TB$, $D = CT^T$, where $F_{i,i+1} = 0$.

Exit 2: The system does not aggregate. $F = TAT^T$, $G = TB$, $D = CT^T$, where

$F_{i,i+1}$ has full column rank.

Upon exit from this algorithm the original system has been transformed into the GHR using orthogonal matrices, a numerically stable transformation process.

The chained aggregation algorithm identifies the supremal A-invariant subspace in the N[C] [13]. This can be understood by referring to Eq. (2.3.2). In the new bases the supremal A-invariant subspace in the N[C] is immediately seen to be

$$\text{sp}\begin{bmatrix}0\\X\end{bmatrix},$$

where $X \in R^{sx1}$, and $s = d(F_{k,k})$.

Any vector lying in this space is obviously A-invariant, i.e.,

$$F\begin{bmatrix}0\\X\end{bmatrix} \subset \begin{bmatrix}0\\X\end{bmatrix}$$

and lies in the N[C],

$$[D_1\ 0\ 0]\begin{bmatrix}0\\0\\X\end{bmatrix} = [0].$$

To better understand Figure 2.1, consider the system at step 3 on the first pass through the algorithm. Aggregation will occur at this stage if and only if the N[C] is A-invariant, i.e., $AN[C] \subset N[C]$. If the F matrix in Figure 2.1 is replaced by the C matrix of the system, the columns of V_2 are seen to span the N[C] and therefore $AV_2 \subset V_2$. This immediately forces the A_{12} submatrix to be zero

$$A_{12} = V_1^T A V_2 \subset V_1^T V_2 = 0 .$$

Thus, by using Figure 2.1 and noting what causes aggregation, an understanding of why the A_{12} submatrix must be zero can be obtained.

2.4. Observability and Controllability Tests

An immediate consequence of the algorithm is its ability to test the observability of a system in a numerically stable manner. By considering the dual problem (A^T , B^T), the controllability of a system can also be checked. Algorithms similar to the chained aggregation algorithm dealing specifically with the controllability problem have been given in [10,12,14].

The following definition is needed.

Definition 2.4.1: The pair (A,C) is said to aggregate, if when represented in the GHR the $F_{k-1,k}$ submatrix equals zero.

The tests using the chained aggregation algorithm may now be given.

Theorem 2.4.2: The following statements are equivalent:

1. The pair (A,C) is unobservable.
2. The system (A,C) will aggregate.

Proof: (1 implies 2).

If the pair (A,C) is unobservable, then the Hautus test will result in a matrix of rank less than n. Since observability is invariant to a state

space transformation, the Hautus test may be applied to the GHR induced by the pair (A,C)

$$\begin{bmatrix} F_{11}-\lambda I & F_{12} & 0 & & \cdots & 0 \\ F_{21} & F_{22}-\lambda I & F_{23} & 0 & \cdots & 0 \\ \vdots & & & & & \vdots \\ F_{k-1,1} & F_{k-1,2} & \cdots & & F_{k-1,k-1}-\lambda I & F_{k-1,k} \\ F_{k,1} & F_{k,2} & \cdots & & F_{k,k-1} & F_{k,k}-\lambda I \\ D_1 & 0 & \cdots & & & 0 \end{bmatrix} .$$

Because of the GHR construction, the only manner in which the above matrix can have rank less than n for any λ is for the $F_{k-1,k}$ submatrix to be zero. This is exactly the condition for aggregation.

(2 implies 1) is now clear. If the system aggregates, then $F_{k-1,k} = 0$ and the pair (A,C) is unobservable, since the rank of the above matrix is less than n for any eigenvalue of the system. □

The dual result follows immediately.

Theorem 2.4.3: The following statements are equivalent:

1. The pair (A,B) is uncontrollable.

2. The system (A^T , B^T) will aggregate.

Proof: Similar to the above proof.

CHAPTER 3

MODIFIED CHAINED AGGREGATION

3.1. Introduction

In the basic chained aggregation algorithm, only the output information structure of the system is used. By incorporating the input structure, the system can be forced to aggregate , using state feedback, when the $\mathbf{R}[B_1]$ contains the $\mathbf{R}[A_{12}]$, where A_{12} and B_1 are the submatrices generated during one of the steps in the algorithm. The ability of the input to satisfy the above condition is determined using the singular value decomposition (SVD) to identify the rows of B_1 which are linearly independent. By the proper choice of input, u, these linearly independent rows can then be used to annihilate the largest possible subblock of the A_{12} submatrix. Further explanation is given below.

This process, called modified chained aggregation (MCA), has been developed in [2] for use in Three-Control-Component-Design (TCCD). The concept of TCCD [2,6,7] arises in the Generalized Hessenberg Representation (GHR) structure. Briefly, this hierarchical design procedure uses the input in three specific ways. First, the input is used to force aggregation. Second, the aggregate dynamics are adjusted to the specifications of the designer. Third, if enough input structure exists the residual dynamics can be adjusted.

The application of the above concepts has been carried over into controller design for a special class of nonlinear systems [5], thus

broadening the scope of problems capable of solution using the developed algorithms.

An algorithm [10] similar to MCA appearing at approximately the same time, although the implementation had not been completed, was motivated by determination of the supremal (A,B)-invariant subspace in the $\mathcal{N}[C]$ [13].

Section 2 demonstrates how the input structure is identified and used to enhance the chained aggregation algorithm. The steps of an implemented algorithm which accomplishes the goals of both procedures above [2,10] are contained in Section 3.

3.2. Identifying The Input Structure

To motivate the use of the input structure within the chained aggregation algorithm, consider the following example.

Suppose the system after one stage of chained aggregation has the following form:

$$\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} u \qquad (3.2.1a)$$

$$y = \begin{bmatrix} D_1 & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} . \qquad (3.2.1b)$$

Aggregation occurs when $A_{12} = 0$. If $A_{12} \neq 0$, but $\mathcal{R}[A_{12}] \subset \mathcal{R}[B_1]$, then by selecting the input $u = - B_1^{\dagger} A_{12} z_2$, the system can be forced to aggregate. This assumes the states z_2 are available for feedback. If these states are

not explicitly available, then some type of dynamic feedback must be introduced to reconstruct them. If $\mathbb{R}[A_{12}] \not\subset \mathbb{R}[B_1]$, then the MCA algorithm identifies the largest subspace of A_{12} contained in the $\mathbb{R}[B_1]$ and performs another step of chained aggregation using the subspace of A_{12} not contained in the $\mathbb{R}[B_1]$ as the aggregating matrix.

SVD is used to calculate the $\mathbb{R}[B_1]$, thus a numerically stable computation is obtained. To identify the $\mathbb{R}[B_1]$ a row compression is performed within the MCA algorithm. The orthogonal transformation which achieves this row compression is then treated as a state space transformation which is used to transform the system matrix. After transforming the system matrix, the A_{12} submatrix is divided into two submatrices; one submatrix $\tilde{A}_{12}$ is not in the $\mathbb{R}[B_1]$, and the other submatrix $\bar{A}_{12}$ is. If $\bar{B}_1$ represents the compressed rows of B_1, then by using the feedback $u = -\bar{B}_1^{\dagger}\bar{A}_{12}\bar{z}_2$, the $\bar{A}_{12}$ submatrix can be annihilated in the system matrix. Again, the states $\bar{z}_2$ have been assumed to be available.

Consider the SVD of the B_1 submatrix

$$B_1 = U \Sigma V^T = \begin{bmatrix} U_1 , U_2 \end{bmatrix} \begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} , \qquad (3.2.2)$$

where $U_1 \in R^{n_1 \times r}$, $U_2 \in R^{n_1 \times (n_1 - r)}$, $S = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$, and the σ_j's are the singular values of B_1. For ease in implementing the algorithm the rows of the B_1 submatrix are compressed down. To achieve this result, B_1 must be premultiplied by $[U_2 , U_1]^T$.

$$\begin{bmatrix} U_2^T \\ U_1^T \end{bmatrix} B_1 = \begin{bmatrix} U_2^T B_1 \\ U_1^T B_1 \end{bmatrix} = \begin{bmatrix} U_2^T \\ U_1^T \end{bmatrix} \begin{bmatrix} U_1 & , & U_2 \end{bmatrix} \begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix}$$

$$= \begin{bmatrix} U_2^T U_1 & U_2^T U_2 \\ U_1^T U_1 & U_1^T U_2 \end{bmatrix} \begin{bmatrix} SV_1^T \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & I_{n_1 - r} \\ I_r & 0 \end{bmatrix} \begin{bmatrix} SV_1^T \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ SV_1^T \end{bmatrix} = \begin{bmatrix} 0 \\ \bar{B}_1 \end{bmatrix} . \qquad (3.2.3)$$

The resulting $\bar{B}_1$ submatrix has full row rank. If the original B matrix had the form $B = [B_1^T , B_2^T]^T$, then the desired row compression in the B_1 submatrix can be carried out by performing a state space transformation with the following matrix.

$$T = \begin{bmatrix} U_2^T & 0 \\ U_1^T & 0 \\ 0 & I_{n-n_1} \end{bmatrix} . \qquad (3.2.4)$$

Performing a state space transformation with this matrix results in the following structure:

$$TAT^T = \begin{bmatrix} U_2^T & 0 \\ U_1^T & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} U_2 & U_1 & 0 \\ 0 & 0 & I \end{bmatrix}$$

$$= \begin{bmatrix} U_2^T A_{11} U_2 & U_2^T A_{11} U_1 & U_2^T A_{12} \\ U_1^T A_{11} U_2 & U_1^T A_{11} U_1 & U_1^T A_{12} \\ A_{12} U_2 & A_{21} U_1 & A_{22} \end{bmatrix} = \begin{bmatrix} A'_{11} & \tilde{A}_{12} \\ A'_{21} & \bar{A}_{12} \\ & A_{22} \end{bmatrix} . \qquad (3.2.5)$$

If $\tilde{A}_{12} \neq 0$, then it is used as the aggregating matrix in the next step of chained aggregation.

The above technique is used throughout the MCA algorithm.

3.3. The Modified Chained Aggregation Algorithm

Initialization: $A^1 = A \in R^{nxn}$, $B^1 = B \in R^{nxm}$, $C \in R^{r_1 xn}$, $\beta = 0$, $\eta = 0$, $\rho = n$, $\tau = 0$, $\alpha = 0$, $i = 1$.

Step 1: Compress the columns of C using SVD

$$C = U^i \Sigma^i V^{iT} .$$

This implies

$$C \begin{bmatrix} V_1^i , V_2^i \end{bmatrix} = \begin{bmatrix} U_1^i , U_2^i \end{bmatrix} \begin{bmatrix} S^i & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} U_1^i S^i , 0 \end{bmatrix} ,$$

where $V_1^i \in R^{nxr_1}$ and $V_2^i \in R^{nx(n-r_1)}$. Let $\alpha = r_1$.

Step 2: Initialize the transformation matrix

$$T^i = V^{iT} .$$

Step 3: Calculate the A_{12}^i submatrix

$$A_{12}^i = V_1^{iT} A^i V_2^i \quad , \quad A_{12}^i \in R^{\alpha x(\rho-\alpha)} .$$

If $A_{12}^i = 0$, then exit 1.

Step 4: Calculate $\hat{B}^i$

$$\hat{B}^i = \begin{bmatrix} I_\tau & 0 \\ 0 & V^{iT} \end{bmatrix} B^i \quad , \quad \hat{B}^i = \begin{bmatrix} 0 \\ \hat{B}_1^i \\ \hat{B}_2^i \end{bmatrix} ,$$

where $0 \in R^{\eta x m}$, $\hat{B}_1^i \in R^{\hat{\beta}_1 x m}$, and $\hat{B}_2^i \in R^{(n-\hat{\beta}_1-\eta) x m}$ with $\hat{\beta}_1 = \alpha + \beta$.

Step 5: Compress the rows of $\hat{B}_1^i$ down using SVD

$$\hat{B}_1^i = \hat{U}^i \hat{\Sigma}^i \hat{V}^{iT} ,$$

$$\begin{bmatrix} \hat{U}_2^{iT} \\ \hat{U}_1^{iT} \end{bmatrix} \hat{B}_1^i = \begin{bmatrix} 0 \\ \hat{U}_1^{iT} \hat{B}_1^i \end{bmatrix} = \begin{bmatrix} 0 \\ \bar{B}_1^i \end{bmatrix} ,$$

where $\bar{B}_1^i \in R^{\beta x m}$.

Note: The value for β has changed between steps 4 and 5. It now equals the number of linearly independent rows in the $\hat{B}_1^i$ submatrix.

Step 6: Update the B matrix

$$B^{i+1} = \begin{bmatrix} 0 \\ 0 \\ \bar{B}_1^i \\ \hat{B}_2^i \end{bmatrix} \begin{matrix} (\eta x m) \\ ((\hat{\beta}_1 - \beta) x m) \\ (\beta x m) \\ ((n - \hat{\beta}_1 - \eta) x m) \end{matrix}$$

Step 7: Update the transformation matrix

$$T^i = \begin{bmatrix} I_\eta & 0 & 0 \\ 0 & \hat{U}_2^{iT} & 0 \\ 0 & \hat{U}_1^{iT} & 0 \\ 0 & 0 & I_{\rho-\alpha} \end{bmatrix} T^i \quad .$$

Step 8: Update the A_{12}^i submatrix

$$\tilde{A}_{12}^i = U_2^{iT} A_{12}^i \quad , \text{ where } \tilde{A}_{12}^i \in R^{(\hat{\beta}_1 - \beta) x (\rho - \alpha)} \quad .$$

$$\bar{A}_{12}^i = U_1^{iT} A_{12}^i \quad , \text{ where } \bar{A}_{12}^i \in R^{\beta x (\rho - \alpha)} \quad .$$

If $\tilde{A}_{12}^i = 0$, then exit 1.

Step 9: Calculate the A_{22}^i submatrix

$$A_{22}^i = V_2^{iT} A^i V_2^i \quad , \text{ where } A_{22}^i \in R^{(\rho-\alpha) x (\rho-\alpha)}$$

$$\rho = \rho - \alpha$$

$$\tau = \tau + \alpha$$

$$i = i + 1 \quad .$$

Step 10: Compress the columns of $\tilde{A}_{12}^{i-1}$

$$\widetilde{A}_{12}^{i-1} = U^i \Sigma^i V^{iT} ,$$

$$\widetilde{A}_{12}^{i-1}\left[V_1^i , V_2^i\right] = \left[U_1^i S^i , 0\right] ,$$

where $V_1^i \in R^{\rho x \alpha}$ and $V_2^i \in R^{\rho x(\rho-\alpha)}$.

If $\alpha = \rho$, then exit 2.

Step 11: Update the transformation matrix

$$T^i = \begin{bmatrix} I_\tau & 0 \\ 0 & V_1^{iT} \\ 0 & V_2^{iT} \end{bmatrix} T^{i-1} .$$

Step 12: Compute the new A_{12}^i submatrix

$$A_{12}^i = \begin{bmatrix} \bar{A}_{12}^{(i-1)} V_2^i \\ V_1^{iT} A_{22}^i V_2^i \end{bmatrix} ,$$

where $A_{12}^i \in R^{(\alpha+\beta)x(\rho-\alpha)}$.

If $A_{12}^i = 0$, then exit 1.

Step 13: $\eta = \tau - \alpha$. Go to Step 4.

Exit 1: The system will aggregate. $F = TAT^T$, $G = TB$, and $D = CT^T$, where $T = T^i$.

Exit 2: The system will not aggregate. $F = TAT^T$, $G = TB$, and $D = CT^T$, where $T = T^i$.

CHAPTER 4

OUTPUT FEEDBACK DESIGN USING THE CHAINED AGGREGATION ALGORITHM

4.1. Introduction

Because of the computational and numerical difficulties associated with large scale systems, a reduced-order model of the system is advantageous during the design process. As stated in Chapter 2, the Generalized Hessenberg Representation (GHR) of a system lends itself nicely to the possibility of identifying a suitable reduced-order model. By comparing the sizes of the submatrices using an appropriately selected norm generated during each stage of chained aggregation, a trade-off between reduced-order model dimension and subsystem coupling can be made to obtain the desired reduced-order model. The resulting reduced-order model contains the strongly observable modes, by construction, which may or may not be the dominant system modes.

This technique of generating a reduced-order model has been conducted on a particular large space structure. The general space structure problem is introduced in Section 2. Section 3 contains the description of the particular structure which was studied. The design procedure which was carried out is detailed in the final section.

4.2. Introduction of the General Large Space Structure Problem

Large Space Structure (LSS) problems have received a great deal of attention. The LSS themselves are usually quite flexible because of weight

restrictions imposed during their transport or deployment in space. The problems are further complicated due to the zero damping environment of space and the light natural damping of the structure itself. A recent review of the literature on LSS Control can be found in [15] with more information available in [16-23].

Initially these systems are often modelled by partial differential equations. For a practical solution, one must reduce the infinite dimensional problem down to one of finite dimension. The most popular method of reducing the infinite dimensional problem has been the finite element method [15], a structural analysis technique. The differential equations resulting from the finite element method are

$$M\ddot{q} + Kq = B_F u \qquad (4.2.1a)$$

$$y = C_v \dot{q} \quad , \qquad (4.2.1b)$$

where $q \in R^n$, $u \in R^m$, $y \in R^r$, and the constant matrices M, K, B_F, and C_v are of compatable dimensions. M is the mass matrix, which in general is positive semidefinite, and K is the stiffness matrix, which is positive definite. The physical nature of most LSS problems allows the damping to be neglected in the modelling, i.e., there are no terms involving $\dot{q}$ in Eq. (4.2.1a).

To perform chained aggregation on the system it must be represented in state space form. A common method to obtain this representation has been to simultaneously diagonalize M and K with a unitary matrix Φ [17,18] such that

$$\Phi^T M \Phi = I \quad \text{and} \quad \Phi^T K \Phi = \Omega^2 .$$

By introducing the transformation $q = \Phi\eta$ Eq. (4.2.1) becomes

$$\ddot{\eta} + \Omega^2 \eta = \Phi^T B_F u \tag{4.2.2a}$$

$$y = C_v \Phi \dot{\eta} \quad . \tag{4.2.2b}$$

A state space model of the system in Eq. (4.2.2) can now be constructed. Let

$$x = \begin{bmatrix} \eta \\ \dot{\eta} \end{bmatrix} .$$

Then Eq. (4.2.2) can be written in state space form.

$$\begin{bmatrix} \dot{\eta} \\ \ddot{\eta} \end{bmatrix} = \begin{bmatrix} 0 & I \\ -\Omega^2 & 0 \end{bmatrix} \begin{bmatrix} \eta \\ \dot{\eta} \end{bmatrix} + \begin{bmatrix} 0 \\ \Phi^T B_F \end{bmatrix} u \quad , \tag{4.2.3a}$$

$$y = [0 \quad C_v \Phi] \begin{bmatrix} \eta \\ \dot{\eta} \end{bmatrix} \quad . \tag{4.2.3b}$$

If M is positive definite then another state space description can be obtained by multiplying through in Eq. (4.2.1) by M^{-1}. Letting

$$x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}$$

yields

$$\begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} 0 & I \\ -M^{-1}K & 0 \end{bmatrix} \begin{bmatrix} q \\ \dot{q} \end{bmatrix} + \begin{bmatrix} 0 \\ M^{-1}B_F \end{bmatrix} u \qquad (4.2.4a)$$

$$y = [0 \quad C_v] \begin{bmatrix} q \\ \dot{q} \end{bmatrix} . \qquad (4.2.4b)$$

The latter state space description was used in the design process because the original coordinates $(q^T, \dot{q}^T)^T$ are retained in the state space description.

For ease in discussion, the following equations are introduced, with the obvious equivalence with Eq. (4.2.4)

$$\dot{x} = Ax + Bu \qquad (4.2.5a)$$

$$y = Cx \quad . \qquad (4.2.5b)$$

4.3. A Physical Problem Description

The structure analyzed herein has been proposed by Charles Stark Draper Labs and can be seen in Figure 4.1. All of the numerical values are summarized in Appendix B. This same structure has been analyzed in [19] using positivity concepts.

The tetrahedral apex represents the antennae feed, with members 1-6 forming the support structure and bi-peds 7-8, 9-10, and 11-12 being supports/controls which are fixed to an inertially stabilized (assumed) antenna dish. The physical nature of the problem allows sensors to be placed only on the bi-peds; no sensors can be placed on the feed itself.

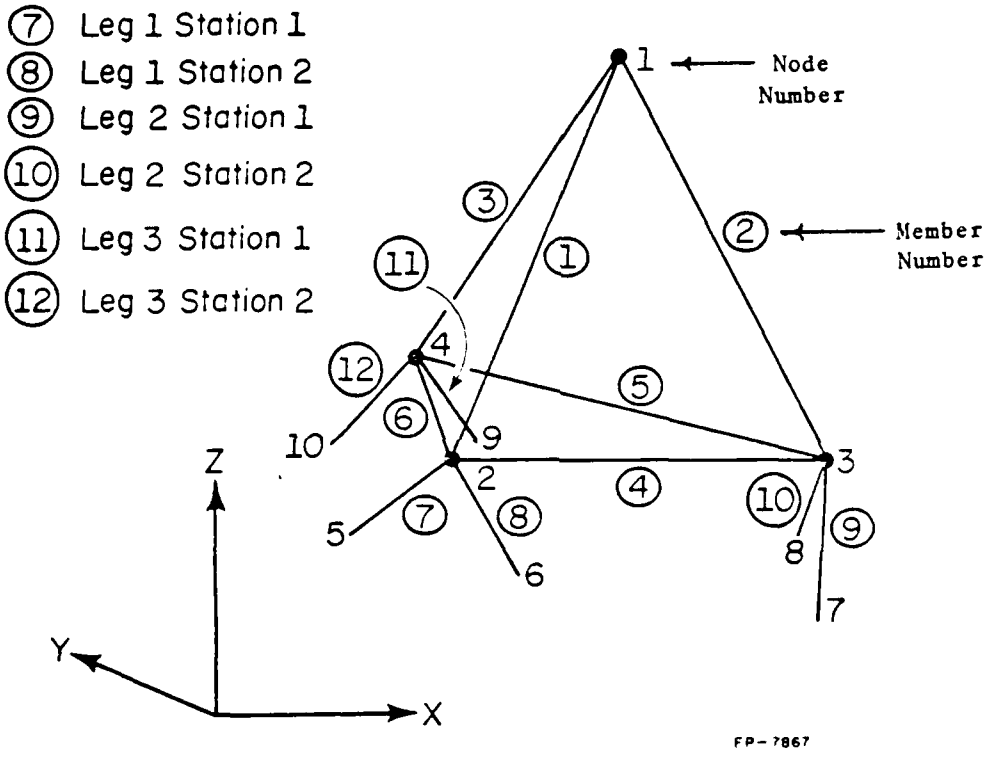


Figure 4.1: Draper Tetrahedral Truss

The actuators control the elongation and contraction of the bi-peds. Only velocity information has been used in the system model; displacement of the bi-peds is not sensed or controlled.

Because uncertainty in the model parameters exists, a perturbed system must be studied. A perturbed system is simply different M and K matrices. For the design to be acceptable, the objective must be satisfied for both the nominal and perturbed systems.

The objective is to damp the x and y deflection of the feed, node 1, to less than .0004 and .00025 units, respectively, in 20 seconds. Thus, the problem is to control the apex in the presence of modelling uncertainty and without directly controlling or measuring its motion or position.

4.4. The Design Procedure

The design process began by verifying the observability and controllability of both the nominal and perturbed systems using the chained aggregation algorithm. The eigenvalues of the two systems were also computed and can be found in Table 4.1.

The next step was to obtain a reduced order model which accurately represented the overall system in general and included the desired x and y deflection modes of node 1 in particular.

The original output structure, the C matrix, of the system contained only velocity measurements of the three bi-peds and by aggregating the system with respect to this information, the x and y deflections of node 1 would be forced into the residual subsystem, an undesirable result. To circumvent this problem a C matrix containing both the controlled outputs,

Table 4.1: Nominal and Perturbed System Eigenvalues

SYSTEM EIGENVALUES				
NOMINAL	0 ± j12.92	0 ± j10.28	0 ± j9.25	0 ± j8.54
	0 ± j4.76	0 ± j4.66	0 ± j4.20	0 ± j3.40
	0 ± j2.96	0 ± j2.89	0 ± j1.66	0 ± j1.34
PERTURBED	0 ± j13.97	0 ± j10.92	0 ± j10.30	0 ± j8.94
	0 ± j5.71	0 ± j5.68	0 ± j5.15	0 ± j3.85
	0 ± j3.56	0 ± j2.96	0 ± j1.47	0 ± j1.17

the x and y displacement of node 1, and the measured outputs, the original, nominal C matrix, was constructed.

Using this C matrix together with the nominal system matrix A, four steps of the chained aggregation algorithm were implemented with the system not aggregating. The transformation matrix resulting from this aggregation process was then used to transform the nominal system, (A,B,C), and the perturbed system, (A^p,B^p,C^p). The following structure for both systems was identified

$$\begin{bmatrix} \dot{z}_a \\ \dot{z}_r \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} z_a \\ z_r \end{bmatrix} + \begin{bmatrix} B_1 \\ 0 \end{bmatrix} u \qquad (4.4.1a)$$

$$y = [C_1 \quad 0] \begin{bmatrix} z_a \\ z_r \end{bmatrix} , \tag{4.4.1b}$$

where $z_a \in R^{16}$, $z_r \in R^8$, $u \in R^6$, $y \in R^6$, and the matrices are partitioned accordingly.

The eigenvalues of the A_{11} and A_{22} submatrices, subsequently referred to as the aggregate and residual subsystems, respectively, were computed and are in Table 4.2. Two immediate conclusions are obtained by comparing the eigenvalues in Table 4.1 with those in Table 4.2. First, the two sets of eigenvalues are both purely imaginary and nearly equal. Second, the lower frequencies were not necessarily placed in the aggregate subsystem, but

Table 4.2: Nominal and Perturbed Subsystem Eigenvalues

SUBSYSTEM EIGENVALUES				
NOMINAL				
AGGREGATE	0 ± j8.04	0 ± j7.43	0 ± j7.43	0 ± j4.63
	0 ± j4.59	0 ± j4.20	0 ± j1.94	0 ± j1.43
RESIDUAL	0 ± j12.00	0 ± j3.52	0 ± j3.02	0 ± j2.96
PERTURBED				
AGGREGATE	0 ± j8.49	0 ± j8.36	0 ± j8.36	0 ± j5.64
	0 ± j5.62	0 ± j5.15	0 ± j1.75	0 ± j1.24
RESIDUAL	0 ± j13.16	0 ± j3.50	0 ± j3.35	0 ± j2.66

instead the strongly observable modes have been forced into the aggregate.

This differs from a modal decomposition of the system where the lower frequency modes are retained to form a reduced order model and the remaining modes are discarded [24]. With the chained aggregation procedure the important modes are those modes which influence the observable modes of the system the most. It is not necessarily concerned with the relative frequencies of the modes.

The effect of the neglected modes, both residual and infinite dimensional, on the reduced order model has been referred to in the literature as controller and observer spillover [17,18]. This refers to how the unmodelled modes are affected by the input and how they affect the output, respectively.

An initial check to see how much coupling existed from the residual subsystem into the aggregate subsystem was performed by comparing the sizes of the submatrices involved. The size used for each submatrix was the matrix 2-norm, i.e., the largest singular value. The norms computed were

$$\| A_{11} \|_2 = 81.92 \quad , \| A_{12} \|_2 = 6.32$$

$$\| A_{21} \|_2 = 94.40 \quad , \| A_{22} \|_2 = 148.37 \quad ,$$

and

$$\| A_{11}^p \|_2 = 94.13 \quad , \| A_{12}^p \|_2 = 5.07$$

$$\| A_{21}^{p} \|_2 = 97.88 \quad , \quad \| A_{22}^{p} \|_2 = 178.18 \quad .$$

The differences in magnitude suggested the coupling was weak and could be neglected [25].

Further analysis supporting this claim was obtained from a geometric setting, following the work in [6] on near unobservability. Intuitively, if the residual subsystem was nearly unobservable in the aggregate subsystem, then the coupling of the residual into the aggregate would be considered weak, i.e., the A_{12} submatrix would have little influence on the aggregate states. As shown later this was made more rigorous by showing that the subspace

$$\mathbf{U} = \mathbf{R}\begin{bmatrix} A_{12} \\ A_{22} \end{bmatrix}$$

was near the subspace

$$\mathbf{V} = \mathbf{R}\begin{bmatrix} 0 \\ A_{22} \end{bmatrix}.$$

This suggested that the residual was nearly unobservable by the aggregate, so the A_{12} submatrix was neglected in the analysis.

To strengthen the concept of one subspace being near another, canonical angles are defined.

Definition 4.4.1 [6,26]: Let $\mathbf{U}$ and $\mathbf{V}$ be subspaces of R^n with orthonormal bases U and V, respectively. Let σ_i be the singular values of $U^T V$. Then the canonical angles between $\mathbf{U}$ and $\mathbf{V}$ are the numbers

$$\theta_i = \cos^{-1} \sigma_i \qquad \square$$

Referring to Figure 2.1 a basis for $\mathbf{U}$ and a basis for $\mathbf{V}$ was obtained by performing singular value decomposition on both matrices $[A_{12}^T, A_{22}^T]^T$ and $[0, A_{22}^T]^T$, separately, and using the generated U_1 submatrix for each basis. The canonical angles between subspaces $\mathbf{U}$ and $\mathbf{V}$ were then obtained using Definition (4.4.1) and can be found in Table 4.3 for both the nominal and perturbed systems. In general, the canonical angles were small and the two subspaces were considered near each other.

Another motivation for neglecting the coupling due to the A_{12} submatrix is because of the structure of the transformed C matrix and the fact that output feedback is being used, i.e., the A_{12} submatrix will not be affected by the feedback.

Table 4.3: Canonical Angles Between $\mathbf{U}$ and $\mathbf{V}$ For The Nominal and Perturbed Systems

CANONICAL ANGLES				
NOMINAL	0.0°	0.0°	0.0°	7.3°
	20.5°	29.9°	34.6°	42.4°
PERTURBED	0.0°	0.0°	12.6°	13.0°
	21.5°	29.9°	34.6°	42.4°

By neglecting the coupling and noting that the residual subsystem was stable, the aggregate subsystem yielded the following reduced order model which was used in the design process.

$$\dot{z}_a = A_{11} z_a + B_1 u \qquad (4.4.2a)$$

$$y = C_1 z_a \quad . \qquad (4.4.2b)$$

This model as well as the reduced order perturbed model were verified to be controllable and observable by using the chained aggregation algorithm.

Because all of the eigenvalues in the aggregate were imaginary, some type of damping had to be introduced. The structure of the transformed nominal input and nominal output matrices suggested that output feedback, $u = -Ky$, be used to introduce the desired damping. The gain matrix, K, was obtained using the procedure outlined in [27,28]; more background material is given in [29,30]. This design approach begins by solving a reference optimal state-feedback linear quadratic regulator problem. The eigenvectors associated with the closed-loop system matrix are then determined. If there are r system outputs, then r of these eigenvectors are retained in the reduced-order output feedback problem, i.e., an r-dimensional eigenspace of the reference problem is retained.

To achieve the desired objective for the x and y deflections of node one, the appropriate Q and R matrices had to be selected. The following analysis led to the desired weighting matrices. The input weightings were selected to be equal and large so that the input energy would not be excessive. The weighting on the states was not as straightforward. After

the necessary transformations were carried out to place the system in the correct structure, the x and y deflections were composed largely of states 15 and 16 with states 13 and 14 corresponding to their respective velocities. Because the x and y deflections as well as their velocities were nearly uncontrollable in the reduced-order model, less weighting than might be expected on these states had to be used. Looking at the feedback structure, states 1-6 were weighted very heavily and states 7-11 were hardly weighted, by comparison. State 12 did require more damping, so it was weighted appropriately. These conclusions were drawn after preliminary trials and observing how the various states related to one another and how their magnitudes compared with the other states. The resulting x and y deflections for both reduced order models can be seen in Figures 4.2 and 4.3.

To verify that the above gain matrix, K, resulted in a desirable design for the full order systems, it was applied to both the nominal and perturbed full order models. The resulting x and y deflections can be seen in Figures 4.4 and 4.5.

By examining the trajectories of the full order systems one can see the slight error introduced by using the reduced-order model during the design procedure. The x deflections are seen to meet the required objective, but the y deflections require 28 and 34 seconds to meet the stricter objective in the nominal and perturbed systems, respectively.

The use of suboptimal dynamic output feedback [27,28,29] could possibly increase the flexibility and robustness of the solution and should be investigated in future work.

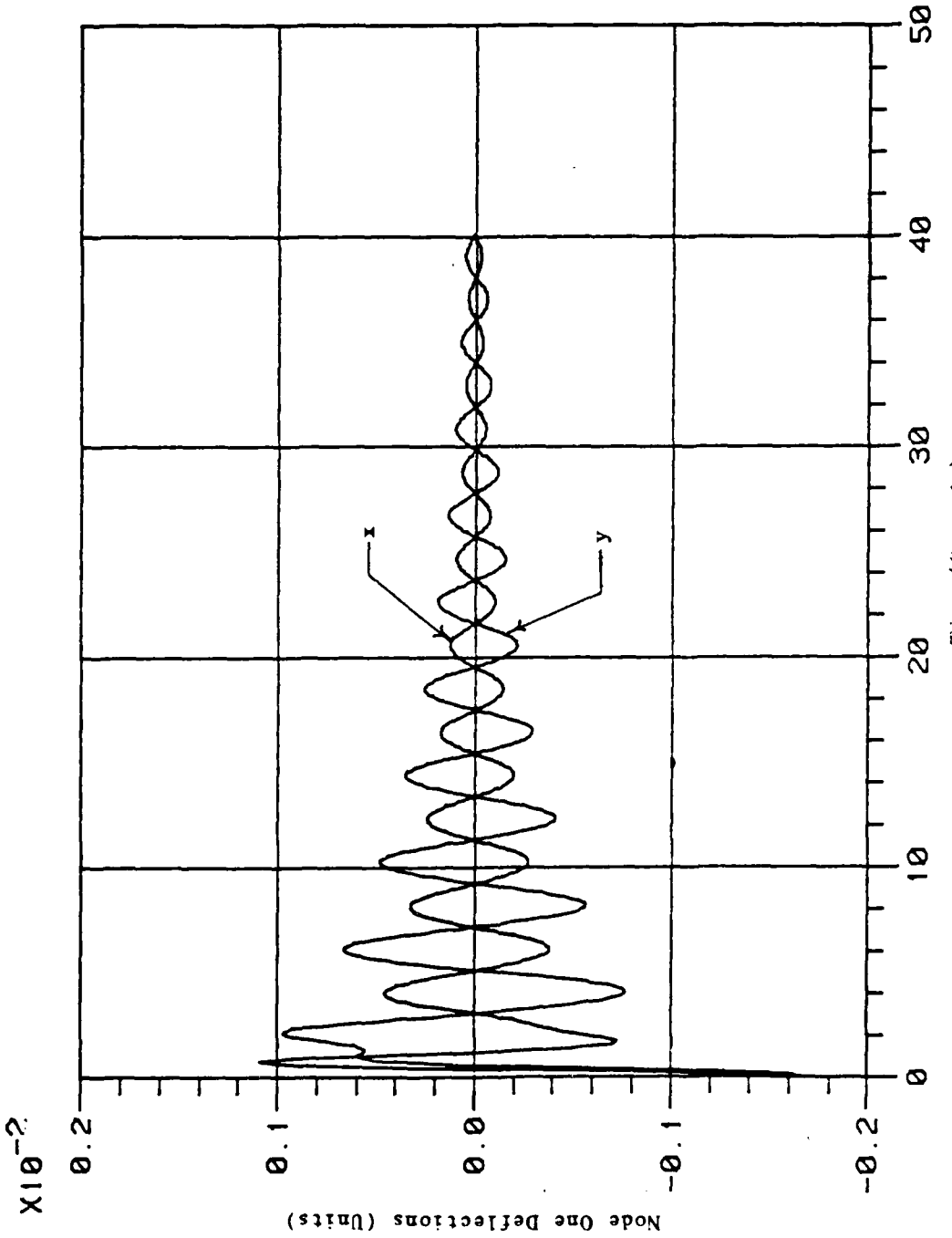


Figure 4.2 Trajectories for the nominal reduced-order model

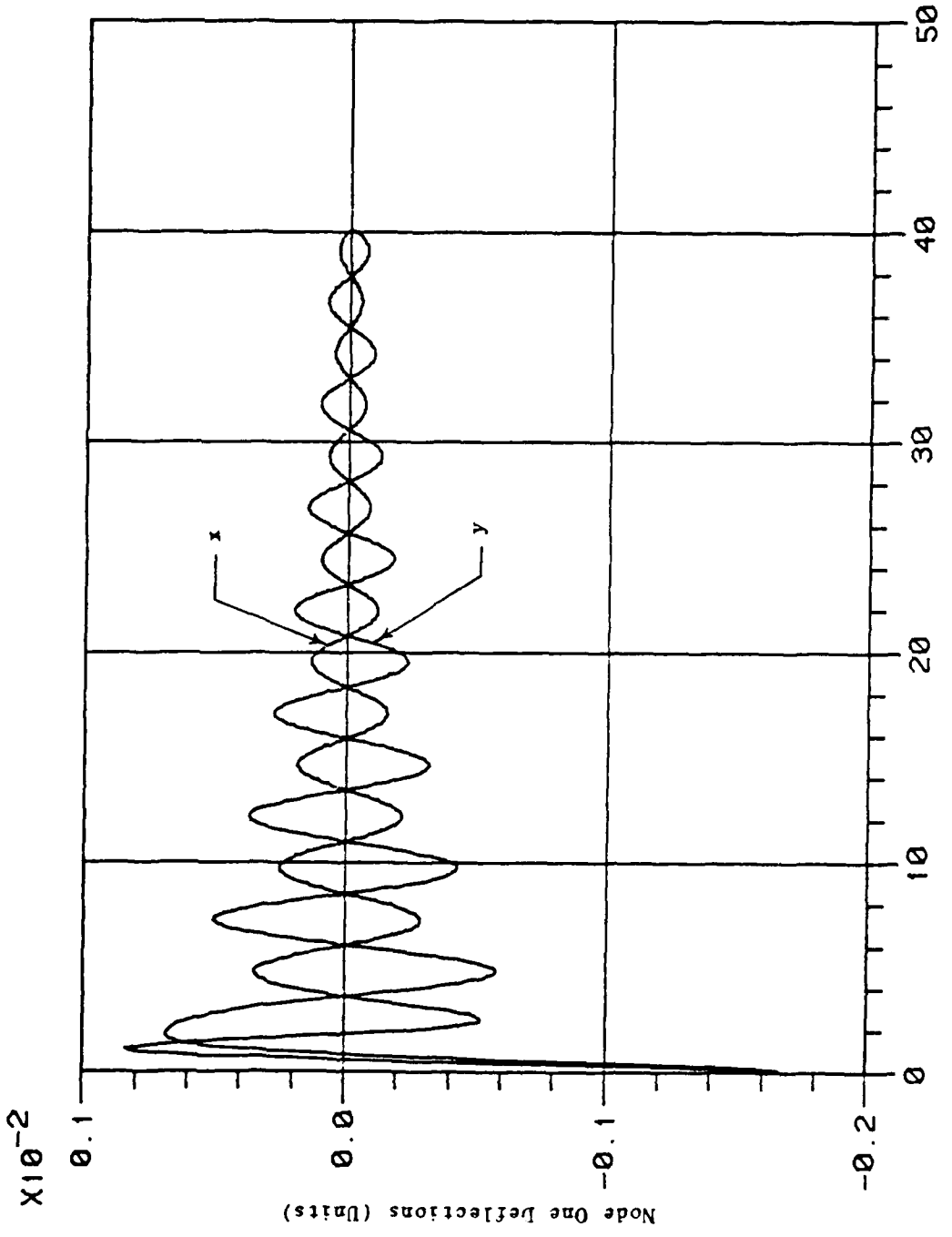


Figure 4.3 Trajectories for the perturbed reduced-order model

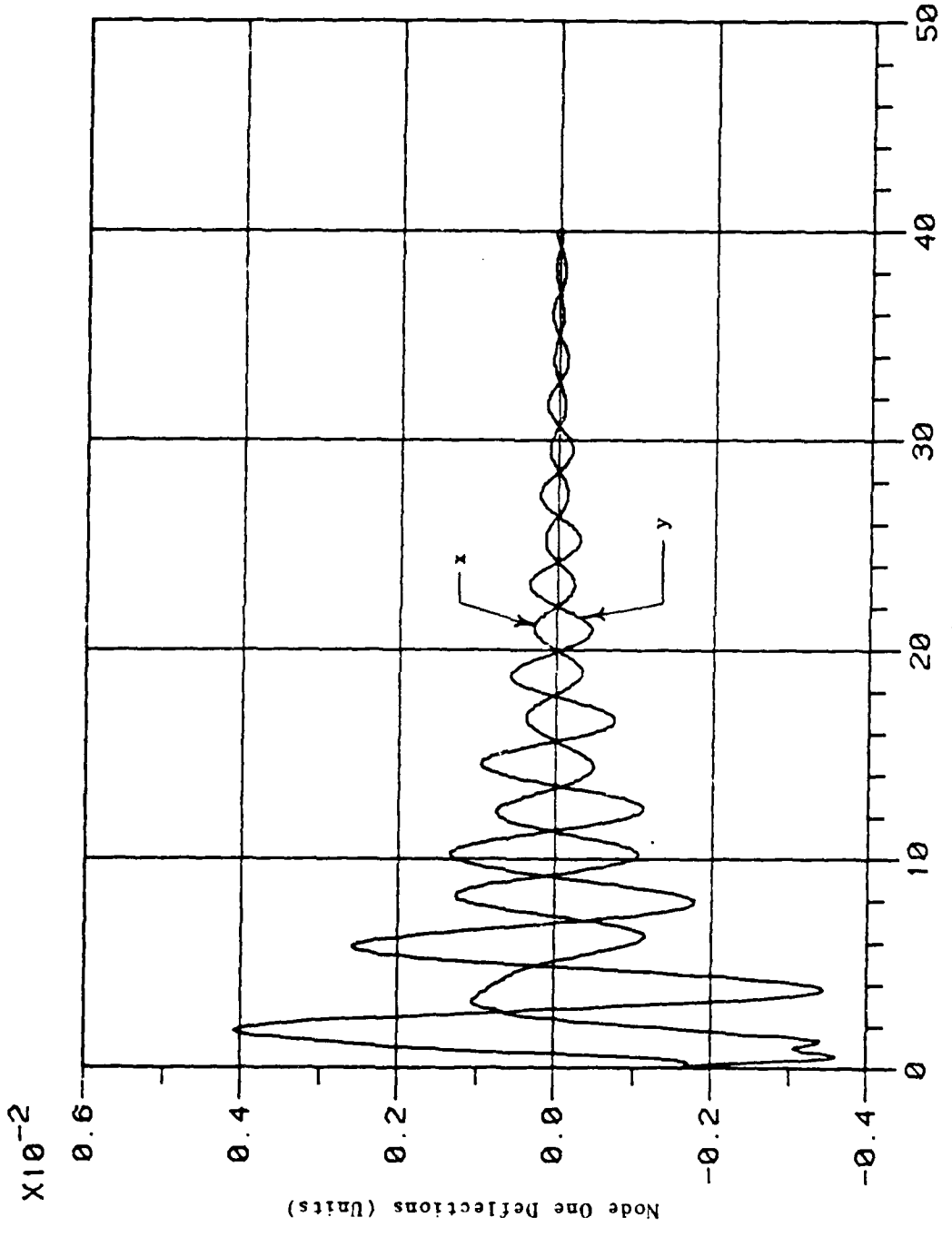


Figure 4.4 Trajectories for the nominal full-order model

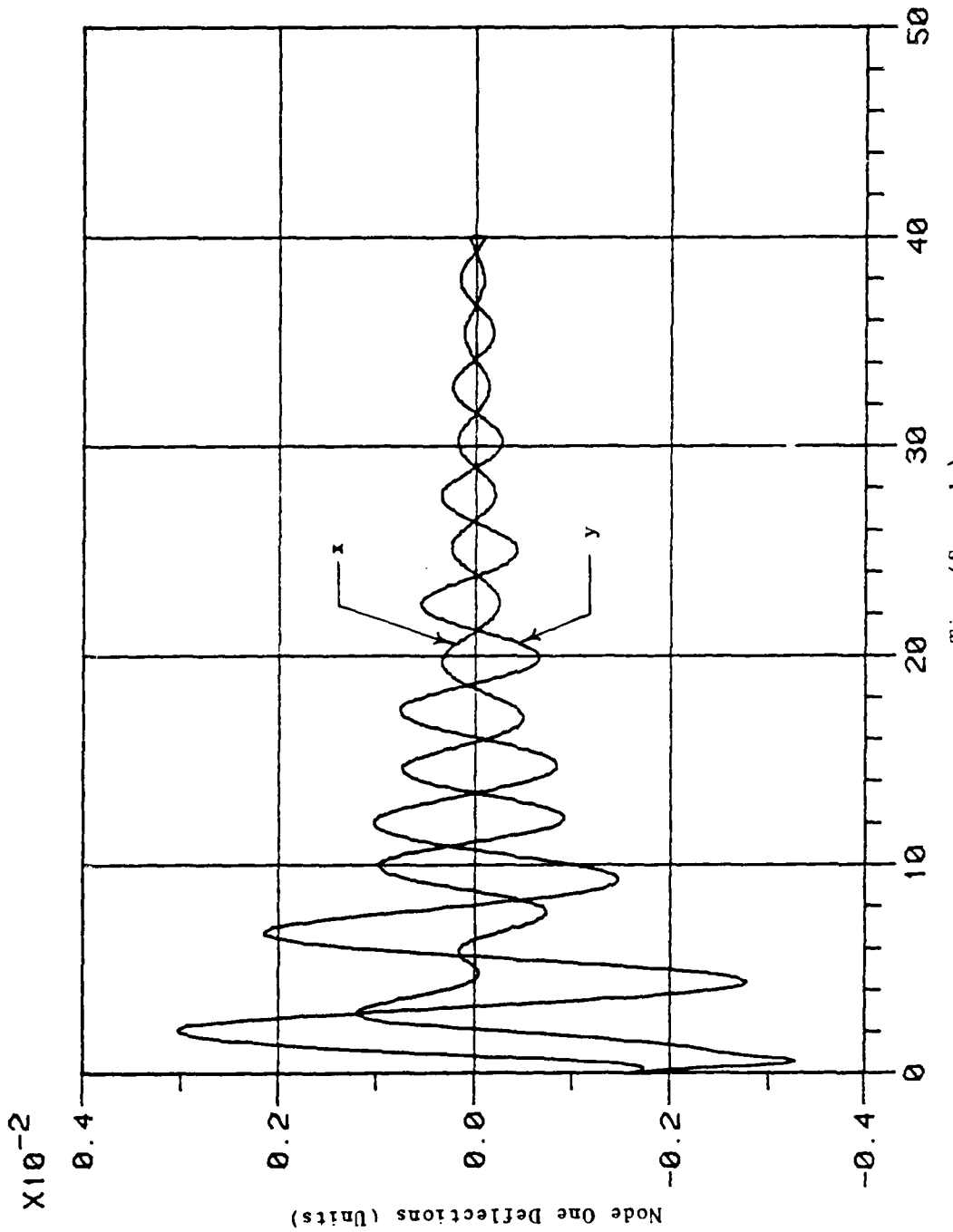


Figure 4.5 Trajectories for the perturbed full-order model

CHAPTER 5

CONCLUSION

A computer implementation of chained aggregation and modified chained aggregation using orthogonal transformation matrices has been presented. To obtain the orthogonal matrices the singular value decomposition has been incorporated into the algorithm whenever a state space transformation must be performed. Because orthogonal matrices have been used, the problem sensitivity will usually not be altered.

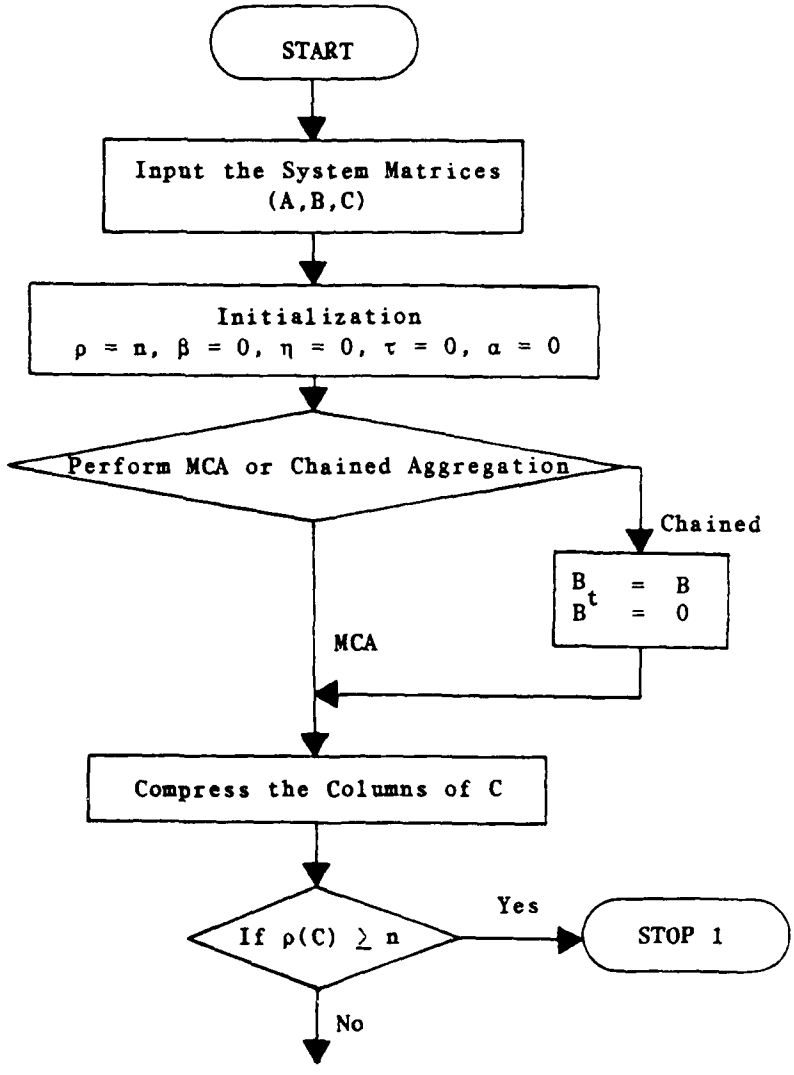
In Chapter 4 the algorithm has been used in the design of a control for a particular large space structure prototype. The reduced-order model suggested by using the chained aggregation algorithm contains the desired information structure, the x and y deflections of node one, and can be used during the design process and, or in the physical implementation of the control, if a dynamical feedback is to be used.

Future research may now focus on the numerical solution of practical problems using the design schemes presented in past Generalized Hessenberg Representation articles.

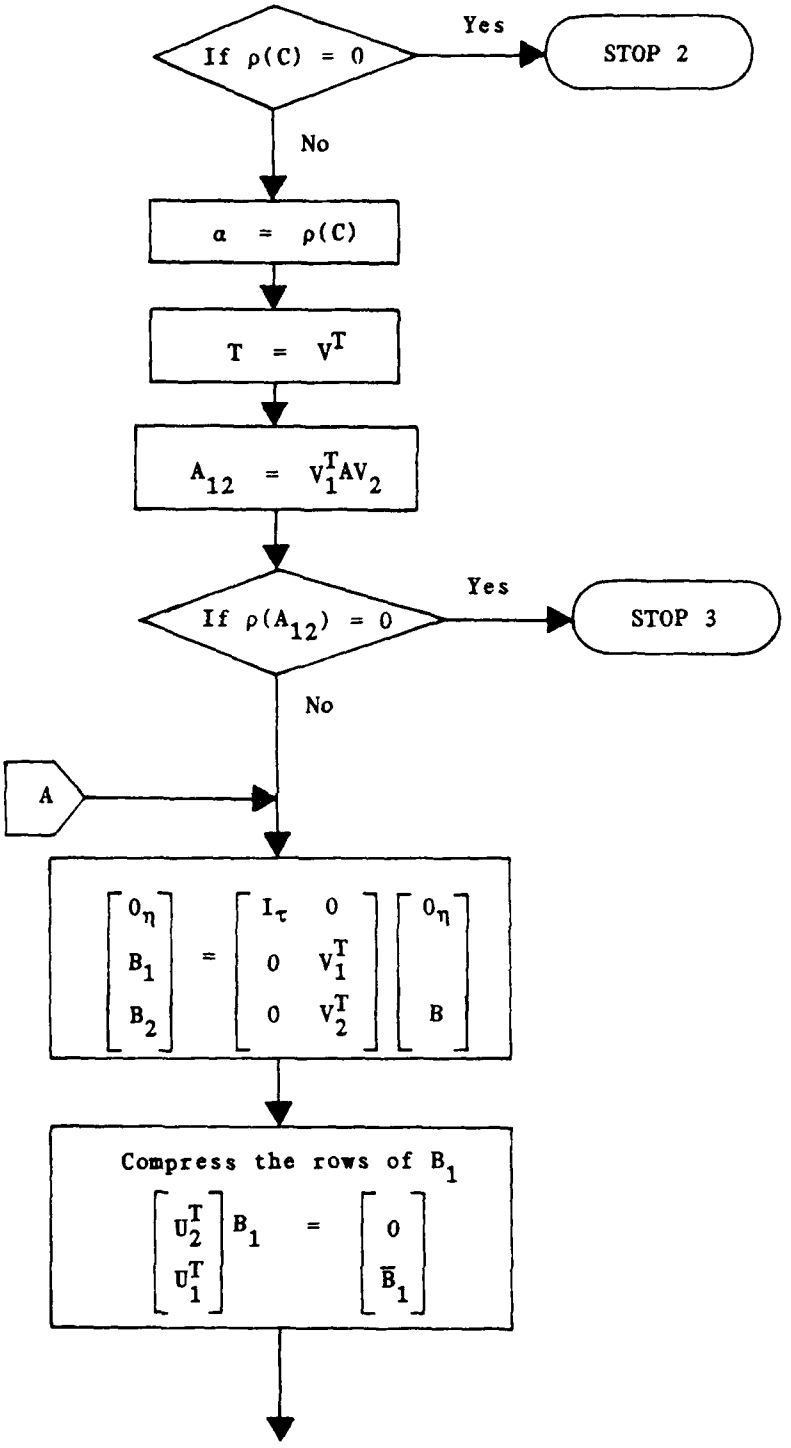
APPENDIX A

SUPPORTING SOFTWARE

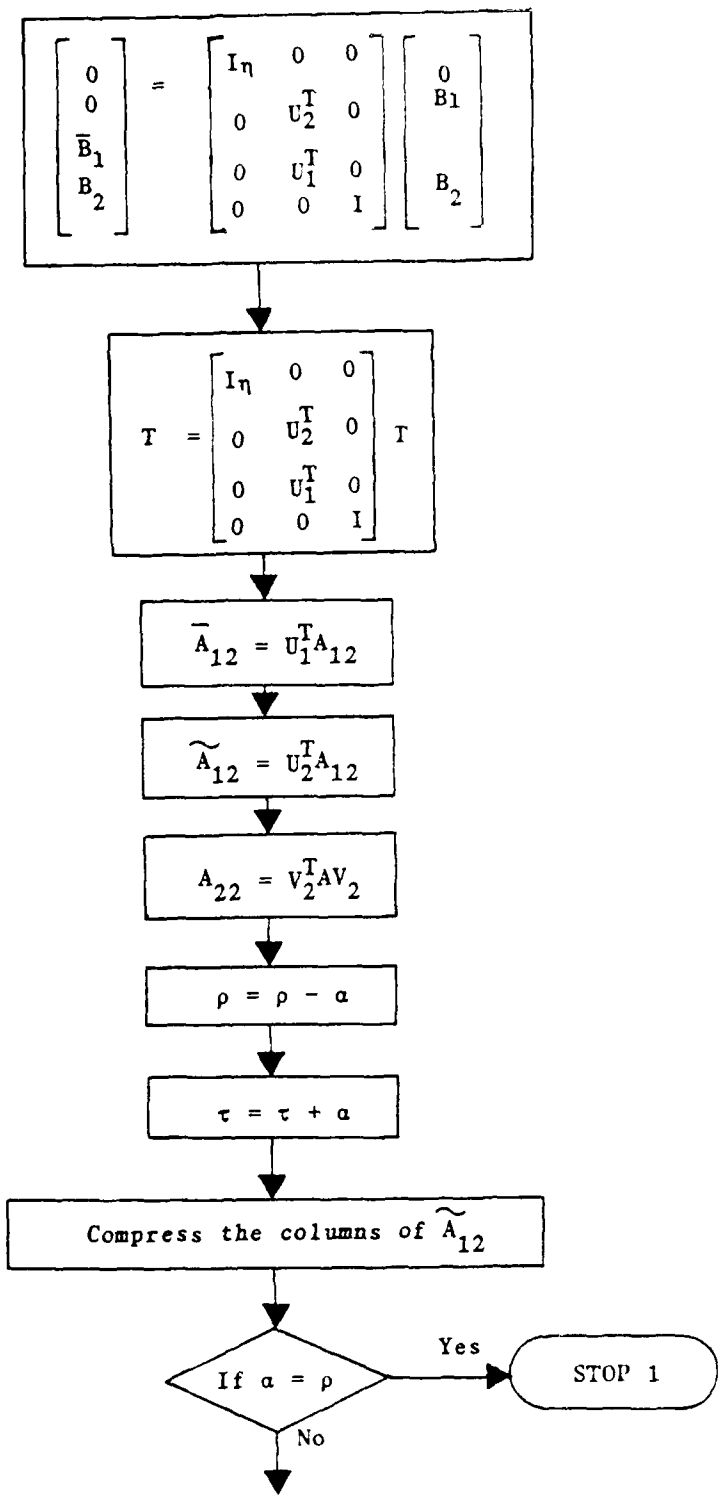
The two algorithms developed in Chapters 2 and 3 have been incorporated into a single computer program. The program has been written in FORTRAN with the singular value decomposition subroutine taken from LINPACK [31]. A flowchart of the software follows.



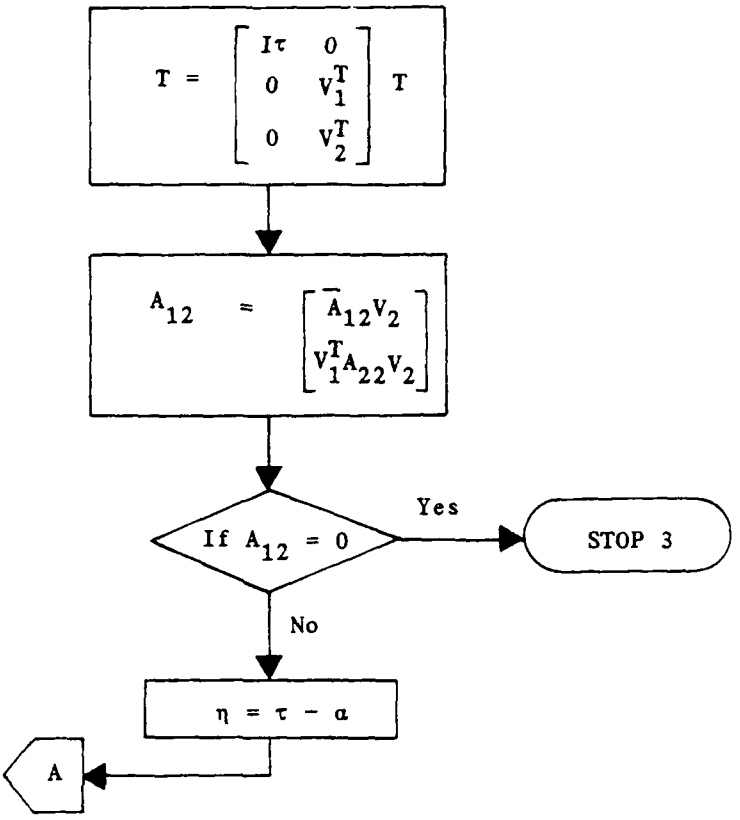
Flowchart continued on next page.



Flowchart continued on next page.



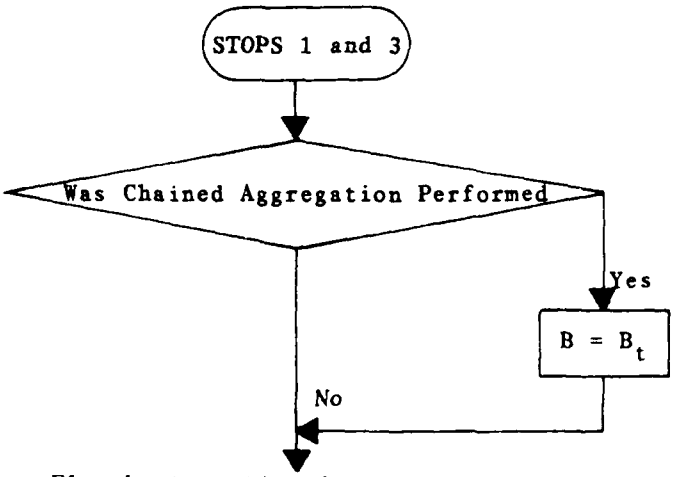
Flowchart continued on next page.



STOP: 1. System will not aggregate.

2. No C matrix.

3. System will aggregate.



Flowchart continued on next page.

$$F = TAT^T,\ G = TB,\ D = CT^T$$

An explanation of the variable names follows:

α = The number of linearly independent columns in the C and A_{12} submatrices. It is redefined after every column compression which is performed on these submatrices.

β = The number of linearly independent rows in the B_1 submatrix. It is redefined after every compression of the rows of B_1.

τ = The sum of the α's as they are generated.

ρ = The size of the current residual subsystem.

η = The number of zero rows identified in the top of the B matrix.

The equivalence between the greek symbols used herein and the variable names used in the actual computer program are as follows.

τ = NIDE

η = NOZERO

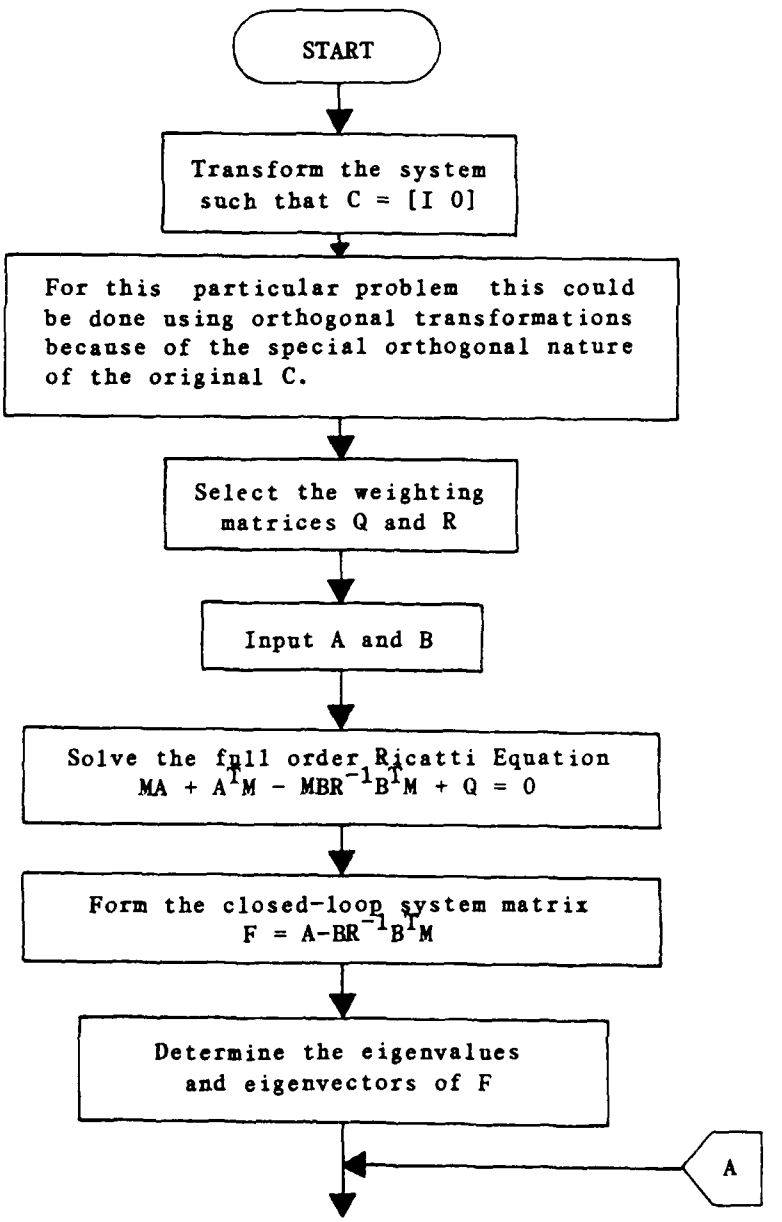
α = NZ

β = NZB

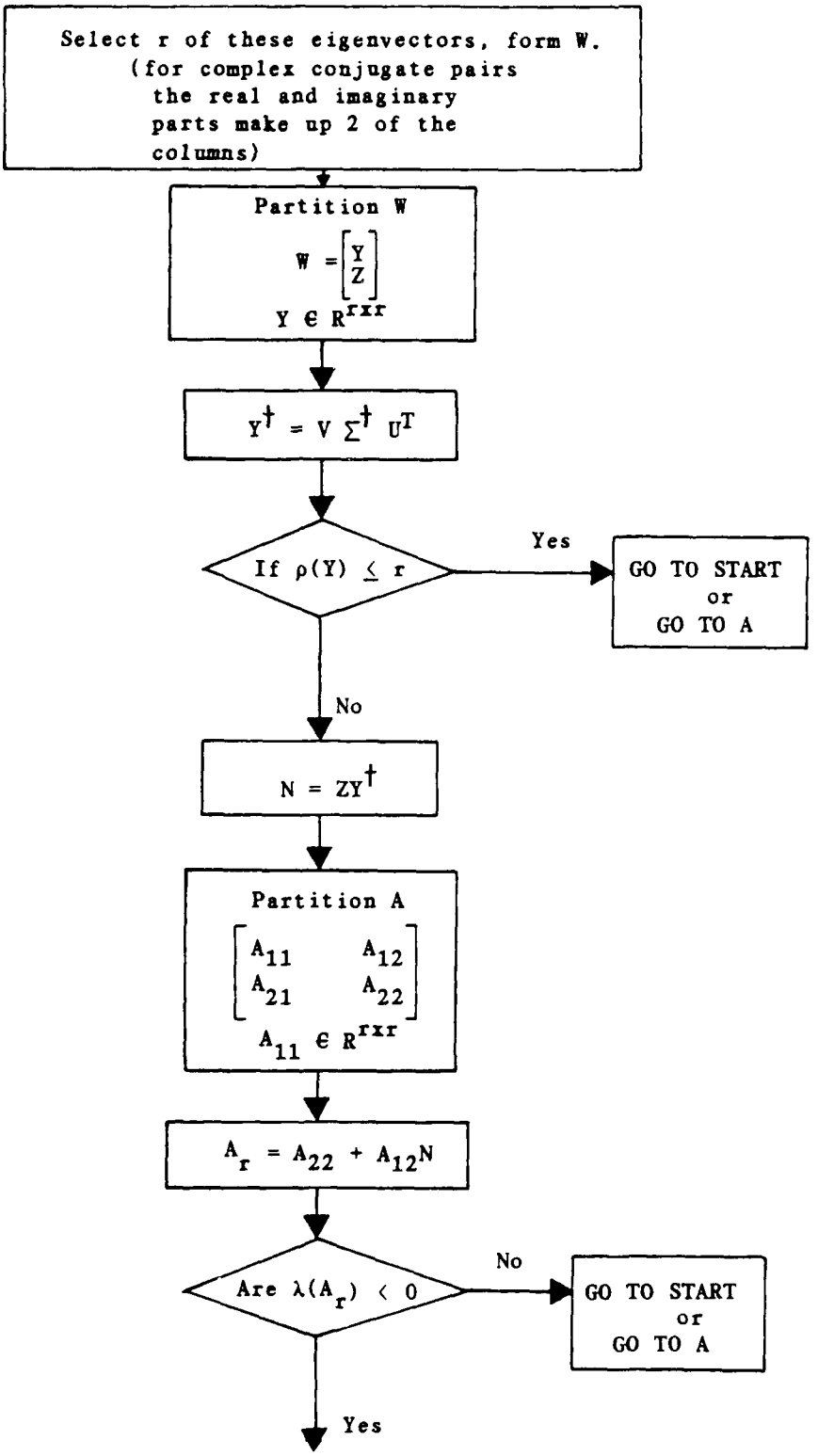
ρ = SIZA

During the design process in Chapter 4 additional software was developed. Many of the routine linear matrix manipulations were performed using LAS [32]. The software required to compute the gain matrix, K, was a

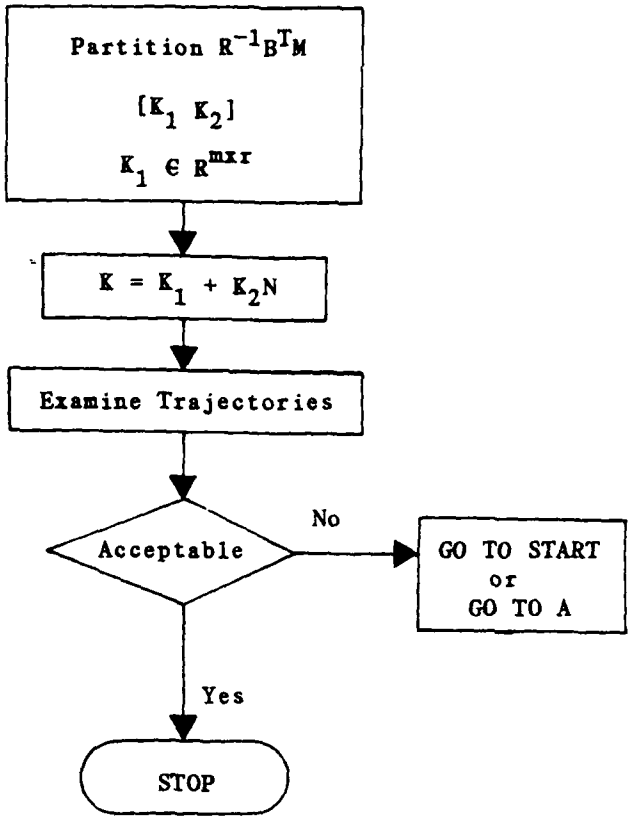
mixture of LAS code and FORTRAN software. The IMSL subroutine EIGRF was used to obtain the eigenvalues and eigenvectors of the closed-loop matrix. A flowchart description of the steps involved follows:



Flowchart continued on next page.



Flowchart continued on next page.



Software independent of LAS was written to generate the system trajectories using the initial condition, the state transition matrix, the total time, and the step size. The state transition matrix was obtained using the e^{At} operator in LAS. A graphing routine which allows the plotting of any of the resulting system trajectories, as well as the x and y deflections of node one was also developed.

APPENDIX B

NUMERICAL VALUES ASSOCIATED WITH THE LARGE SPACE STRUCTURE

The numerical values associated with Eq. (4.2.1) for both the nominal and perturbed systems are given in Tables B.1,B.2,B.3,B.4 and Tables B.5,B.6, respectively. Note that matrices B_F and C_v are identical for both systems. If the state space representation in Eq. (4.2.3) is desired, the required $\bar{a}$ matrices are given in Tables B.7 and B.8. The relationship between the coordinates $(q^T , \dot{q}^T)^T$ and the nodes of Figure 4.1 can be found in Table B.9. The transformation matrix used to generate Eq. (4.4.1) for both the nominal and perturbed systems is given in Table B.10. All of the reduced-order system matrices (A_{11},B_1,C_1) and (A_{11}^p,B_1^p,C_1^p) are contained in Tables B.11,B.12,B.13 and Tables B.14,B.15, and B.16, respectively. To obtain the system used during the design of the K feedback matrix, the transformation matrix in Table B.17 was used. Tables B.18 and B.19 contain the final Q and R matrices used to generate the K feedback matrix which is given in Table B.20.

Table B.1: Mass Matrix for the Nominal System

$$M = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

Table B.2: Stiffness Matrix for the Nominal System

$$K = \begin{bmatrix}
27.50 & 12.99 & 36.74 & -25.00 & -14.43 & -40.82 & -2.50 & 1.44 & 4.08 & .00 & .00 & .00 \\
12.99 & 12.50 & 21.21 & -14.43 & -8.33 & -23.57 & 1.44 & -.83 & -2.36 & .00 & -3.33 & 4.71 \\
36.74 & 21.21 & 80.00 & -40.82 & -23.57 & -66.67 & 4.08 & -2.36 & -6.67 & .00 & 4.71 & -6.67 \\
-25.00 & -14.43 & -40.82 & 158.84 & 42.43 & 40.82 & -100.00 & .00 & .00 & -25.00 & -43.30 & .00 \\
-14.43 & -8.33 & -23.57 & 42.43 & 109.85 & 23.57 & .00 & .00 & .00 & -43.30 & -75.00 & .00 \\
-40.82 & -23.57 & -66.67 & 40.82 & 23.57 & 102.02 & .00 & .00 & .00 & .00 & .00 & .00 \\
-2.50 & 1.44 & 4.08 & -100.00 & .00 & .00 & 136.34 & -29.43 & -4.08 & -25.00 & 43.30 & .00 \\
1.44 & -.83 & -2.36 & .00 & .00 & .00 & -29.43 & 102.35 & 2.36 & 43.30 & -75.00 & .00 \\
4.08 & -2.36 & -6.67 & .00 & .00 & .00 & -4.08 & 2.36 & 42.02 & .00 & .00 & .00 \\
.00 & .00 & .00 & -25.00 & -43.30 & .00 & -25.00 & 43.30 & .00 & 85.35 & .00 & .00 \\
.00 & -3.33 & 4.71 & -43.30 & -75.00 & .00 & 43.30 & -75.00 & .00 & .00 & 153.33 & -4.71 \\
.00 & 4.71 & -6.67 & .00 & .00 & .00 & .00 & .00 & .00 & .00 & -4.71 & 42.02
\end{bmatrix}$$

Table B.3: The B_F Matrix

$$B_F = \begin{bmatrix} .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\ .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\ .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\ -.3535 & .3536 & .0000 & .0000 & .0000 & .0000 \\ .6124 & -.6123 & .0000 & .0000 & .0000 & .0000 \\ -.7071 & -.7071 & .0000 & .0000 & .0000 & .0000 \\ .0000 & .0000 & -.3536 & .3535 & .0000 & .0000 \\ .0000 & .0000 & -.6123 & .6124 & .0000 & .0000 \\ .0000 & .0000 & -.7071 & -.7071 & .0000 & .0000 \\ .0000 & .0000 & .0000 & .0000 & .7071 & -.7071 \\ .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\ .0000 & .0000 & .0000 & .0000 & -.7071 & -.7071 \end{bmatrix}$$

Table B.4: The C_v Matrix

$$C_v = \begin{bmatrix} .0000 & .0000 & .0000 & -.3535 & .6124 & -.7071 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\ .0000 & .0000 & .0000 & .3536 & -.6123 & -.7071 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\ .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & -.3536 & -.6123 & -.7071 & .0000 & .0000 & .0000 \\ .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .3535 & .6124 & -.7071 & .0000 & .0000 & .0000 \\ .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .7071 & .0000 & -.7071 \\ .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & -.7071 & .0000 & -.7071 \end{bmatrix}$$

Table B.5: Mass Matrix for the Perturbed System

$$M = \begin{bmatrix} 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

Table B.6: Stiffness Matrix for the Perturbed System

$$K = \begin{bmatrix}
33.75 & 15.16 & 42.87 & -30.00 & -17.32 & -48.99 & -3.75 & 2.17 & 6.12 & .00 & .00 & .00 \\
15.16 & 16.25 & 24.75 & -17.32 & -10.00 & -28.29 & 2.17 & -1.25 & -3.54 & .00 & -5.00 & 7.07 \\
42.87 & 24.75 & 100.00 & -48.99 & -28.29 & -80.00 & 6.12 & -3.54 & -10.00 & .00 & 7.07 & -10.00 \\
-30.00 & -17.32 & -48.99 & 193.26 & 46.32 & 48.98 & -120.00 & .00 & .00 & -30.00 & -51.96 & .00 \\
-17.32 & -10.00 & -28.29 & 46.32 & 139.78 & 28.29 & .00 & .00 & .00 & -51.96 & -90.00 & .00 \\
-48.99 & -28.29 & -80.00 & 48.98 & 28.29 & 133.03 & .00 & .00 & .00 & .00 & .00 & .00 \\
-3.75 & 2.17 & 6.12 & -120.00 & .00 & .00 & 167.01 & -31.16 & -6.12 & -30.00 & 51.96 & .00 \\
2.17 & -1.25 & -3.54 & .00 & .00 & .00 & -31.16 & 131.02 & 3.54 & 51.96 & -90.00 & .00 \\
6.12 & -3.54 & -10.00 & .00 & .00 & .00 & -6.12 & 3.54 & 63.03 & .00 & .00 & .00 \\
.00 & .00 & .00 & -30.00 & -51.96 & .00 & -30.00 & 51.96 & .00 & 113.03 & .00 & .00 \\
.00 & -5.00 & 7.07 & -51.96 & -90.00 & .00 & 51.96 & -90.00 & .00 & .00 & 185.00 & -7.07 \\
.00 & 7.07 & -10.00 & .00 & .00 & .00 & .00 & .00 & .00 & .00 & -7.07 & 63.03
\end{bmatrix}$$

Table B.7: $\hat{A}$ Matrix for the Nominal System

$$\hat{A} = \begin{bmatrix}
-.3444 & .5429 & -.0492 & .0573 & -.1369 & .0000 & .0557 & -.0758 & .1445 & -.0058 & .1594 & .0837 \\
.5964 & .3135 & -.0284 & -.0992 & -.0791 & .0000 & -.0965 & -.0438 & .0835 & .0100 & .0921 & .0483 \\
.0000 & -.1990 & .3788 & -.0001 & -.3441 & .0000 & .0000 & .1837 & .2702 & .0000 & .2580 & .1587 \\
-.0311 & .1263 & .3125 & -.1760 & .1621 & -.2041 & -.0344 & .0470 & .2125 & -.2242 & -.1517 & -.4059 \\
.0538 & .0729 & .1805 & .3046 & .0936 & .3535 & .0596 & .0272 & .1228 & .3883 & -.0876 & -.2343 \\
.0000 & .0976 & .0652 & .0000 & -.4969 & .0000 & .0000 & .0978 & -.3266 & .0000 & -.3117 & -.1611 \\
-.0508 & .1098 & .2727 & -.2388 & .1620 & -.2041 & -.0288 & .0368 & -.1414 & .3846 & -.1619 & .2996 \\
.0652 & .0416 & .1274 & .3409 & .0731 & -.3535 & .0564 & .0325 & -.3096 & .0368 & .3311 & -.1449 \\
.0638 & -.0673 & .1371 & -.0916 & -.0757 & .0001 & .4873 & -.4696 & -.0150 & -.0118 & .0009 & -.0082 \\
-.0311 & .0909 & .2466 & -.1759 & .1444 & .4082 & -.0345 & .0466 & -.3389 & -.2241 & .2058 & .0269 \\
.0756 & .0742 & .1726 & .3771 & .1037 & .0000 & .0532 & .0157 & .0323 & -.3147 & -.3058 & .3304 \\
-.0638 & -.0673 & .1372 & .0915 & -.0757 & .0000 & -.4872 & -.4698 & -.0150 & .0119 & .0009 & -.0082
\end{bmatrix}$$

Table B.8: $\hat{A}$ Matrix for the Perturbed System

$$\hat{A} = \begin{bmatrix}
-.2471 & .3999 & .0637 & .0275 & -.0878 & .0000 & -.0266 & -.0299 & .0991 & -.0034 & .0637 & .0321 \\
.4279 & .2309 & .0368 & -.0476 & -.0507 & .0000 & .0461 & -.0173 & .0572 & .0058 & .0368 & .0185 \\
.0000 & -.1489 & .4000 & .0000 & -.1299 & .0000 & .0000 & .0878 & .1729 & .0000 & .0959 & .0644 \\
-.0196 & .0833 & .1984 & -.1718 & .3095 & -.2041 & .0337 & .0407 & .1076 & -.2286 & -.2401 & -.4026 \\
.0398 & .0481 & .1145 & .2977 & .1786 & .3535 & -.0584 & .0236 & .0621 & .3960 & -.1386 & -.2324 \\
.0000 & .0681 & .2010 & .0000 & -.3514 & .0000 & .0000 & .0355 & -.4953 & .0000 & -.2605 & -.1305 \\
-.0361 & .0700 & .1548 & -.2512 & .2866 & -.2041 & .0273 & .0274 & -.1679 & .3783 & -.0863 & .3204 \\
.0435 & .0225 & .0680 & .3436 & .1224 & -.3535 & -.0548 & .0280 & -.2198 & .0455 & .3944 & -.1587 \\
.0440 & -.0472 & .0978 & -.0819 & .0114 & .0001 & -.4913 & -.4875 & -.0111 & -.0147 & .0070 & -.0090 \\
-.0196 & .0545 & .1363 & -.1718 & .2494 & .4082 & .0338 & .0380 & -.2743 & -.2286 & .2984 & .0227 \\
.0530 & .0494 & .1000 & .3894 & .1868 & .0000 & -.0511 & .0098 & -.0355 & -.3049 & -.2719 & .3568 \\
-.0440 & -.0472 & .0978 & .0819 & .0114 & .0000 & .4909 & -.4879 & -.0111 & .0147 & .0070 & -.0090
\end{bmatrix}$$

Table B.9: Relationship Between the Coordinates and the Nodes

q_1 ($\dot{q}_1$) $\equiv$ Displacement (Velocity) of node 1 in the negative x direction.

q_2 ($\dot{q}_2$) $\equiv$ Displacement (Velocity) of node 1 in the negative y direction.

q_3 ($\dot{q}_3$) $\equiv$ Displacement (Velocity) of node 1 in the negative z direction.

q_4 ($\dot{q}_4$) $\equiv$ Displacement (Velocity) of node 2 in the negative x direction.

q_5 ($\dot{q}_5$) $\equiv$ Displacement (Velocity) of node 2 in the negative y direction.

q_6 ($\dot{q}_6$) $\equiv$ Displacement (Velocity) of node 2 in the negative z direction.

q_7 ($\dot{q}_7$) $\equiv$ Displacement (Velocity) of node 3 in the negative x direction.

q_8 ($\dot{q}_8$) $\equiv$ Displacement (Velocity) of node 3 in the negative y direction.

q_9 ($\dot{q}_9$) $\equiv$ Displacement (Velocity) of node 3 in the negative z direction.

q_{10} ($\dot{q}_{10}$) $\equiv$ Displacement (Velocity) of node 4 in the negative x direction.

q_{11} ($\dot{q}_{11}$) $\equiv$ Displacement (Velocity) of node 4 in the negative y direction.

q_{12} ($\dot{q}_{12}$) $\equiv$ Displacement (Velocity) of node 4 in the negative z direction.

Table B.10: Transformation matrix which generates Eq. (4.4.1)
$T = [\ T_1,\ T_2,\ T_3\ ]$

$$T_1 = \begin{bmatrix}
1.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & 1.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .3104 & -.4680 & -.2701 & -.4736 & .0660 & .4352 \\
.0000 & .0000 & -.0002 & -.3374 & .5846 & .0003 & .5133 & .0934 \\
.0000 & .0000 & .4170 & .0379 & .0222 & -.6324 & -.0673 & -.4554 \\
.0000 & .0000 & .1492 & .0171 & .0101 & .1074 & .0730 & .0017 \\
.0000 & .0000 & -.0001 & -.0236 & .0406 & .0000 & -.0325 & .0460 \\
.0000 & .0000 & .0000 & -.2887 & .5000 & .0000 & -.2887 & -.5000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .5646 & -.4162 & -.2401 & .5909 & -.1369 & -.1567 \\
.0000 & .0000 & -.0003 & -.2285 & .3958 & 0002 & -.3900 & .4890 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & -.0044 & -.0058 & -.0034 & .0002 & .0076 & -.0036 \\
.0000 & .0000 & -.2904 & -.3933 & -.2272 & .0200 & .5403 & -.2494 \\
.0000 & .0000 & -.5517 & -.4487 & -.2591 & -.1211 & -.4185 & -.1283
\end{bmatrix}$$

Table B.10: (Continued)

$$T_2 = \begin{bmatrix}
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & -.2192 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & -.4494 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
-.0030 & .4097 & -.1606 & -.0030 & .0000 & .0000 & .0000 & .0000 \\
-.0049 & -.3374 & -.3979 & .0049 & .0000 & .0000 & .0000 & .0000 \\
-.0118 & -.4283 & .1691 & -.0118 & .0000 & .0000 & .0000 & .0000 \\
-.6905 & .0381 & .0624 & -.6916 & .0000 & .0000 & .0000 & .0000 \\
.7046 & -.0237 & .0513 & -.7036 & .0000 & .0000 & .0000 & .0000 \\
.0003 & .5773 & -.0001 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & -.9996 & -.0288 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0288 & -.9996 & .0000 & .0000 \\
.0851 & -.2043 & -.0398 & .0852 & .0000 & .0000 & .0000 & .0000 \\
-.0653 & -.2285 & .5824 & .0652 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & -.6203 & .6515 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0002 & .0001 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & -.6875 & -.3336 \\
.0002 & .0007 & .0084 & .0002 & .0000 & .0000 & .3776 & .4628 \\
.0200 & .0543 & .5924 & .0203 & .0000 & .0000 & -.0055 & -.0067 \\
-.1210 & -.3203 & -.2984 & -.1210 & .0000 & .0000 & -.0001 & -.0002
\end{bmatrix}$$

Table B.10: (Continued)

T_3 =

.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
.0000	.0000	-.2188	-.3792	.8991	.0000	.0000	.0000
.3797	-.8988	.0000	.0000	.0000	.0000	.0000	.0000
.0000	.0000	.0000	.0000	.0000	.7071	.0000	-.7071
.0000	.0000	.0000	.0000	.0000	-.7071	.0000	-.7071
.7783	.4384	.0000	.0000	.0000	.0000	.0000	.0000
.0000	.0000	-.4496	-.7786	-.4378	.0000	.0000	.0000
.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
.0000	.0000	.0000	.0000	.0000	.0C00	.0000	.0000
.3761	.0001	.1360	-.0785	.0000	.0000	.1569	.0000
.0000	.0000	.6124	-.3536	-.0001	.0000	-.7070	.0000
-.1926	.0000	-.3769	.2176	.0000	.0000	-.4355	.0000
.2672	.0001	-.4630	.2673	.0001	.0000	-.5346	.0000
-.0039	.0000	.0067	-.0039	.0000	.0000	.0077	.0000
-.0001	.0000	.0002	-.0001	.0000	.0000	.0002	.0000

Table B.11: Nominal reduced-order system matrix
$A = [\, A_1, A_2 \,]$

$$A_1 = \begin{bmatrix}
.00 & .00 & .00 & .00 & .00 & .00 & .00 & .00 \\
.00 & .00 & .00 & .00 & .00 & .00 & .00 & .00 \\
-1.83 & 1.06 & .00 & .00 & .00 & .00 & .00 & .00 \\
-18.34 & -10.59 & .00 & .00 & .00 & .00 & .00 & .00 \\
.00 & 1.67 & .00 & .00 & .00 & .00 & .00 & .00 \\
.00 & 1.67 & .00 & .00 & .00 & .00 & .00 & .00 \\
8.95 & 5.17 & .00 & .00 & .00 & .00 & .00 & .00 \\
.89 & -.52 & .00 & .00 & .00 & .00 & .00 & .00 \\
.00 & .00 & -.18 & .43 & .29 & -.29 & -.21 & -.37 \\
.00 & .00 & -.15 & .30 & -.24 & .24 & .61 & -.30 \\
.00 & .00 & .18 & .57 & -.29 & .31 & -.28 & .39 \\
.00 & .00 & -.64 & -.10 & .52 & .46 & .05 & .27 \\
.00 & .00 & .62 & .02 & .48 & .51 & .04 & -.33 \\
.00 & .00 & .25 & .25 & .41 & -.41 & .52 & .52 \\
13.93 & 6.67 & .00 & .00 & .00 & .00 & .00 & .00 \\
6.10 & 6.06 & .00 & .00 & .00 & .00 & .00 & .00
\end{bmatrix}$$

Table B.11: (Continued)

$$A_2 = \begin{bmatrix}
.00 & .00 & .00 & .00 & .00 & .00 & -1.00 & .03 \\
.00 & .00 & .00 & .00 & .00 & .00 & -.03 & -1.00 \\
17.19 & 12.86 & -10.53 & 13.76 & -13.33 & -4.47 & .00 & .00 \\
-42.45 & -23.86 & -40.56 & .87 & -.17 & -4.48 & .00 & .00 \\
-27.22 & 19.99 & 18.54 & -10.96 & -10.40 & -7.22 & .00 & .00 \\
26.20 & -18.52 & -21.41 & -10.50 & -10.67 & 7.22 & .00 & .00 \\
20.70 & -48.97 & 19.77 & -.43 & -.34 & -9.17 & .00 & .00 \\
33.66 & 24.04 & -26.28 & -6.35 & 6.71 & -9.17 & .00 & .00 \\
.00 & .00 & .00 & .00 & .00 & .00 & .00 & .00 \\
.00 & .00 & .00 & .00 & .00 & .00 & .00 & .00 \\
.00 & .00 & .00 & .00 & .00 & .00 & .00 & .00 \\
.00 & .00 & .00 & .00 & .00 & .00 & .00 & .00 \\
.00 & .00 & .00 & .00 & .00 & .00 & .00 & .00 \\
.00 & .00 & .00 & .00 & .00 & .00 & .00 & .00 \\
23.77 & -.57 & 19.98 & -1.26 & 1.44 & .00 & .00 & .00 \\
12.83 & 1.02 & 10.78 & -.68 & -2.66 & .00 & .00 & .00
\end{bmatrix}$$

Table B.12: Input matrix for the nominal reduced-order model

$$B = \begin{bmatrix}
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & -.1631 & -.4726 & .0000 & .0000 \\
.4728 & .1628 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .5000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .5000 \\
.1627 & -.4728 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .4726 & -.1631 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000
\end{bmatrix}$$

Table B.13: Output matrix for the nominal reduced-order model

$$C = [\, C_1,\ C_2 \,]$$

$$C_1 = \begin{bmatrix} .0000 & .0000 & .0000 & .9455 & .0000 & .0000 & .3255 & .0000 \\ .0000 & .0000 & .0000 & .3255 & .0000 & .0000 & -.9455 & .0000 \\ .0000 & .0000 & -.3262 & .0000 & .0000 & .0000 & .0000 & .9453 \\ .0000 & .0000 & -.9453 & .0000 & .0000 & .0000 & .0000 & -.3262 \\ .0000 & .0000 & .0000 & .0000 & 1.0000 & .0000 & .0000 & .0000 \\ .0000 & .0000 & .0000 & .0000 & .0000 & 1.0000 & .0000 & .0000 \end{bmatrix}$$

$$C_2 = \begin{bmatrix} .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\ .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\ .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\ .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\ .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\ .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \end{bmatrix}$$

Table B.14: Perturbed reduced-order system matrix
$A^p = [\ A_1,\ A_2\ ]$

$$A_1 = \begin{bmatrix}
.00 & .00 & .00 & .00 & .00 & .00 & .00 & .00 \\
.00 & .00 & .00 & .00 & .00 & .00 & .00 & .00 \\
-2.75 & 1.59 & .00 & .00 & .00 & .00 & .00 & .00 \\
-22.01 & -12.71 & .00 & .00 & .00 & .00 & .00 & .00 \\
.00 & 2.50 & .00 & .00 & .00 & .00 & .00 & .00 \\
.00 & 2.50 & .00 & .00 & .00 & .00 & .00 & .00 \\
10.74 & 6.20 & .00 & .00 & .00 & .00 & .00 & .00 \\
1.34 & -.77 & .00 & .00 & .00 & .00 & .00 & .00 \\
.00 & .00 & -.18 & .43 & .29 & -.29 & -.21 & -.37 \\
.00 & .00 & -.15 & .30 & -.24 & .24 & .61 & -.30 \\
.00 & .00 & .18 & .57 & -.29 & .31 & -.28 & .39 \\
.00 & .00 & -.64 & -.10 & .52 & .46 & .05 & .27 \\
.00 & .00 & .62 & .02 & .48 & .51 & .04 & -.33 \\
.00 & .00 & .25 & .25 & .41 & -.41 & .52 & .52 \\
8.54 & 3.91 & .00 & .00 & .00 & .00 & .00 & .00 \\
3.55 & 3.95 & .00 & .00 & .00 & .00 & .00 & .00
\end{bmatrix}$$

Table B.14: (Continued)

$$A_2 = \begin{bmatrix}
.00 & .00 & .00 & .00 & .00 & .00 & -1.00 & .03 \\
.00 & .00 & .00 & .00 & .00 & .00 & -.03 & -1.00 \\
21.77 & 16.49 & -13.08 & 20.69 & -19.97 & -6.71 & .00 & .00 \\
-53.20 & -30.20 & -51.69 & 1.56 & -.31 & -6.71 & .00 & .00 \\
-34.35 & 25.48 & 23.42 & -16.51 & -15.55 & -10.83 & .00 & .00 \\
32.83 & -23.27 & -27.74 & -15.68 & -16.06 & 10.83 & .00 & .00 \\
25.94 & -61.98 & 25.19 & -.76 & -.63 & -13.76 & .00 & .00 \\
42.25 & 30.32 & -33.84 & -9.43 & 10.12 & -13.76 & .00 & .00 \\
.00 & .00 & .00 & .00 & .00 & .00 & .00 & .00 \\
.00 & .00 & .00 & .00 & .00 & .00 & .00 & .00 \\
.00 & .00 & .00 & .00 & .00 & .00 & .00 & .00 \\
.00 & .00 & .00 & .00 & .00 & .00 & .00 & .00 \\
.00 & .00 & .00 & .00 & .00 & .00 & .00 & .00 \\
.00 & .00 & .00 & .00 & .00 & .00 & .00 & .00 \\
14.20 & -.42 & 11.82 & -1.03 & 1.08 & .00 & .00 & .00 \\
7.66 & .77 & 6.38 & -.56 & -2.00 & .00 & .00 & .00
\end{bmatrix}$$

Table B.15: Input matrix for the perturbed reduced-order model

$$B^p = \begin{bmatrix}
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & -.1631 & -.4726 & .0000 & .0000 \\
.4728 & .1628 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .5000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .5000 \\
.1627 & -.4728 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .4726 & -.1631 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000
\end{bmatrix}$$

Table B.16: Output matrix for the perturbed reduced-order model

$$C^p = [\ C_1,\ C_2\ ]$$

$$C_1 = \begin{bmatrix} .0000 & .0000 & .0000 & .9455 & .0000 & .0000 & .3255 & .0000 \\ .0000 & .0000 & .0000 & .3255 & .0000 & .0000 & -.9455 & .0000 \\ .0000 & .0000 & -.3262 & .0000 & .0000 & .0000 & .0000 & .9453 \\ .0000 & .0000 & -.9453 & .0000 & .0000 & .0000 & .0000 & -.3262 \\ .0000 & .0000 & .0000 & .0000 & 1.0000 & .0000 & .0000 & .0000 \\ .0000 & .0000 & .0000 & .0000 & .0000 & 1.0000 & .0000 & .0000 \end{bmatrix}$$

$$C_2 = \begin{bmatrix} .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\ .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\ .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\ .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\ .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\ .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \end{bmatrix}$$

Table B.17: Transformation used during the design of the feedback matrix K
$T = [\ T_1,\ T_2\ ]$

$$T_1 = \begin{bmatrix}
.0000 & .0000 & .0000 & .9455 & .0000 & .0000 & .3255 & .0000 \\
.0000 & .0000 & .0000 & .3255 & .0000 & .0000 & -.9455 & .0000 \\
.0000 & .0000 & -.3262 & .0000 & .0000 & .0000 & .0000 & .9453 \\
.0000 & .0000 & -.9453 & .0000 & .0000 & .0000 & .0000 & -.3262 \\
.0000 & .0000 & .0000 & .0000 & 1.0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & 1.0000 & .0000 & .0000 \\
.2198 & .1269 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
-.0006 & .0008 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
-.1990 & -.1149 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0950 & .0549 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
-.0678 & .1173 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
-.0001 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
-.9082 & -.2308 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
-.2715 & .9490 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000
\end{bmatrix}$$

Table B.17: (Continued)

$$T_2 = \begin{bmatrix}
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.9549 & -.0001 & .1541 & -.0030 & .0000 & .0000 & .0000 & .0000 \\
.0002 & 1.0000 & -.0001 & .0000 & .0000 & .0000 & .0000 & .0000 \\
.2143 & -.0002 & -.9493 & .0039 & .0000 & .0000 & .0000 & .0000 \\
-.0264 & .0000 & -.0366 & -.9929 & .0001 & .0000 & .0000 & .0000 \\
.0000 & -.0001 & .0000 & -.0001 & -.9908 & .0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & -1.0000 & .0000 & .0000 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & -.8801 & -.4748 \\
.0000 & .0000 & .0000 & .0000 & .0000 & .0000 & -.4748 & .8801 \\
.1970 & -.0003 & .2623 & -.1146 & .0348 & .0001 & .0000 & .0000 \\
-.0523 & -.0009 & -.0696 & .0304 & .1310 & .0001 & .0000 & .0000
\end{bmatrix}$$

Table B.18: State weighting matrix Q

$$
Q = \begin{bmatrix}
10^6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 10^6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 10^6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 10^6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 10^6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 10^6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 10^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 10^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10^2 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10^2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10^4 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10^4 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10^4 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10^4 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10^4
\end{bmatrix}
$$

Table B.19: Input weighting matrix R

$$R = \begin{bmatrix} 10^3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10^3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10^3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10^3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10^3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10^3 \end{bmatrix}$$

Table B.20: Output feedback gain matrix K

$$K = \begin{bmatrix} 31.6249 & .0024 & .0087 & -.0134 & -.0140 & .0086 \\ .0024 & 31.6249 & .0086 & -.0140 & -.0134 & .0087 \\ -.0039 & -.0025 & 31.6224 & .0006 & .0011 & -.0005 \\ .0125 & .0134 & -.0002 & 31.6228 & .0006 & -.0013 \\ .0134 & .0125 & -.0013 & .0006 & 31.6228 & -.0002 \\ -.0025 & -.0039 & -.0005 & .0011 & .0006 & 31.6224 \end{bmatrix}$$

APPENDIX C

PROGRAM LISTING

```
      CHARACTER*1          ANS     , CFRMT   , AFRMT   , BFRMT
      CHARACTER*20         FANAME  , FCNAME  , FBNAME  , NAME
      CHARACTER*20         INAME
      INTEGER              I       , J       , NCC     , NRC
      INTEGER              NM      , NMIN    , IERR    , NOZERO
      INTEGER              FA12    , NIDE    , NZ      , SOS
      INTEGER              NRB     , NCB     , SIZA    , NZB
      INTEGER              K       , ID      , JD      , NRGDMB
      INTEGER              NMINB   , FA12S   , NRA12S  , STEP
      INTEGER              FCHND   , NCBTMP
      REAL    WORK(100)
      REAL    A(100,100)   ,  B(100,100)     ,  G(100,100)
      REAL    C(100,100)   ,  A12(100,100)   ,  TA12(100,100)
      REAL    U(100,100)   ,  V(100,100)     ,  T(100,100)
      REAL    E(100)       ,  GDMB(100,100)  ,  TDMB(100,100)
      REAL    A12B(100,100), A12S(100,100)   ,  DMB(100,100)
      REAL    NORM         , SIZCHK
      REAL    SIGMA(100)
      LOGICAL              INFO
*
*****************************************************************
*  VERSION aggregate.f CREATED DECEMBER 22,1982  BY: HAL THARP  *
*****************************************************************
*
*  TO COMPILE THIS PROGRAM:
*  f77 -u -o agg aggregate.f
*
************************************
*  VARIABLE TABLE:  MAIN PROGRAM  *
************************************
*
*  A       :  CONTAINS THE A MATRIX IN THE STATE SPACE DESCRIPTION.
*
*  ANS     :  A CHARACTER VARIABLE WHICH IS USED TO CONTAIN THE
*             RESPONSE OF THE USER TO A QUESTION.
*
*  A12     :  CONTAINS THE CURRENT A12 SUBMATRIX.
*
*  A12B    :  CONTAINS THE SUBMATRIX RESULTING FROM TRANSFORMING
*             A12 WITH PART OF THE U MATRIX GENERATED DURING THE
*             ROW COMPRESSION OF A B SUBMATRIX.  THIS MATRIX IS
*             CONTAINED IN THE RANGE OF THE INPUT.
*
*  A12S    :  CONTAINS THE SUBMATRIX RESULTING FROM TRANSFORMING
*             A12 WITH PART OF THE U MATRIX GENERATED DURING THE
```

```
*            ROW COMPRESSION OF A B SUBMATRIX.  THIS MATRIX IS
*            NOT CONTAINED IN THE RANGE OF THE INPUT.
*
*  B       :  CONTAINS THE B MATRIX IN THE STATE SPACE DESCRIPTION.
*
*  C       :  CONTAINS THE C MATRIX IN THE STATE SPACE DESCRIPTION.
*
*  DMB     :  A DUPLICATE OF GDMB WHICH IS PASSED TO THE SSVDC
*             SUBROUTINE.  THIS MUST BE DONE BECAUSE THE MATRIX
*             RETURNED FROM SSVDC IS ALTERED FROM THE ORIGINAL
*             MATRIX PASSED.
*
*  E       :  VECTOR THAT ORDINARILY CONTAINS ZEROES.  IT IS USED
*             IN THE SSVDC ROUTINE.  FOR MORE INFORMATION CONSULT
*             LINPACK USERS' GUIDE, DONGARRA, et.al. CHPT 11.
*
*  FA12    :  A FLAG WHICH IS ZERO IF THE A12 SUBMATRIX IS ZERO
*             AND ONE OTHERWISE.
*
*  FA12S   :  A FLAG WHICH IS ZERO IF THE A12S SUBMATRIX IS ZERO
*             AND ONE OTHERWISE.
*
*  FCHND   :  A FLAG WHICH IS SET TO 1 WHEN CHAINED AGGREGATION IS
*             PERFORMED AND 0 WHEN MODIFIED CHAINED AGGREGATION IS
*             PERFORMED.
*
*  F_NAME  :  CONTAINS THE NAME OF THE FILE WHERE THE (A,B,C) MATRICES
*             ARE STORED.
*
*  _FRMT   :  CONTAINS THE FORMAT METHOD USED IN THE STORAGE OF THE
*             PARTICULAR (A,B,C) MATRIX.  a,A = SPARSE   OR  b,B =
*             NORMAL.
*
*  G       :  THE ARRAY USED TO STORE THE SUBMATRIX GENERATED DURING
*             THE TRANSFORMATION OF THE B MATRIX.  THIS SUBMATRIX IS
*             THEN PLACED IN THE CORRECT LOCATION WITHIN B.
*
*  GDMB    :  A TEMPORARY ARRAY WHICH CONTAINS THE SUBMATRIX OF THE
*             B MATRIX WHOSE ROWS ARE TO BE COMPRESSED.
*
*  IERR    :  VARIABLE CONTAINING ZERO WHEN THE SINGULAR VALUES HAVE
*             BEEN COMPUTED CORRECTLY.  SEE LINPACK USERS' GUIDE,
*             CHPT. 11 WHEN IERR IS NOT EQUAL TO ZERO.
*
*  INAME   :  CHARACTER VARIABLE WHICH IS THE NAME OF THE FILE
*             CONTAINING THE STAGE INFORMATION.  IT IS NOT USED IF
*             INFO IS FALSE.
*
*  INFO    :  A LOGICAL VARIABLE WHICH IS TRUE WHEN THE USER REQUESTS
*             THE STAGE INFORMATION TO BE OUTPUT INTO A FILE.
*
*  I,J,K   :  INTEGER VARIABLES USED WITHIN DO LOOPS AS THE COUNTERS.
*  ID,JD
```

```
*
*  NAME     :  A CHARACTER VARIABLE USED TO IDENTIFY THE PARTICULAR
*              MATRIX BEING INPUT INTO THE PROGRAM.
*
*  NCBTMP   :  AN INTEGER VARIABLE USED TO STORE THE NUMBER OF
*              COLUMNS IN THE ORIGINAL B MATRIX WHEN CHAINED AGGREGATION
*              IS PERFORMED.
*
*  NC_      :  CONSTANT CONTAINING THE COLUMN DIMENSION OF THE PARTICULAR
*              MATRIX (B,C,ETC.).
*
*  NIDE     :  KEEPS TRACK OF THE NUMBER OF NON-ZERO SINGULAR VALUES
*              GENERATED BY COMPRESSING THE C MATRIX AND THE A12S
*              SUBMATRICES.  THIS IS THE NUMBER OF IDENTITY ELEMENTS
*              NEEDED IN THE UPPER LEFT HAND CORNER WHEN UPDATING THE
*              TRANSFORMATION MATRIX.
*
*  NM       :  CONSTANT CONTAINING THE MAXIMUM POSSIBLE DIMENSION OF THE
*              MATRICES PASSED TO THE SINGULAR VALUE DECOMPOSITION
*              ROUTINE (SSVDC).  CURRENTLY SET EQUAL TO 100.
*
*  NMIN     :  AN INTEGER VARIABLE CONTAINING THE MINIMUM OF THE NUMBER
*              OF COLUMNS OR THE NUMBER OF ROWS IN THE C MATRIX OR AN
*              A12S SUBMATRIX.
*
*  NMINB    :  AN INTEGER VARIABLE CONTAINING THE MINIMUM OF THE NUMBER
*              OF COLUMNS OR THE NUMBER OF ROWS OF THE B SUBMATRIX WHICH
*              IS PASSED TO SSVDC.  THIS IS THE MAXIMUM NUMBER OF NON-
*              ZERO SINGULAR VALUES POSSIBLE FOR THE PARTICULAR SUBMATRIX
*
*  NORM     :  CONSTANT CONTAINING THE NORM OF THE ORIGINAL A MATRIX.
*
*  NOZERO   :  KEEPS TRACK OF THE NUMBER OF ZERO ROWS IN THE INPUT
*              MATRIX.
*
*  NR_      :  CONSTANT CONTAINING THE ROW DIMENSION OF THE PARTICULAR
*              MATRIX (B,C,ETC.).
*
*  NZ       :  AN INTEGER CONTAINING THE NUMBER OF NON-ZERO SINGULAR
*              VALUES RESULTING FROM A DECOMPOSITION OF THE C MATRIX
*              OR AN A12 SUBMATRIX.
*
*  NZB      :  NUMBER OF NON-ZERO SINGULAR VALUES OBTAINED AFTER PASSING
*              PORTIONS OF THE B MATRIX TO THE SSVDC ROUTINE.
*
*  SIGMA    :  VECTOR CONTAINING THE SINGULAR VALUES RESULTING FROM THE
*              SSVDC ROUTINE.
*
*  SIZA     :  CONTAINS THE DIMENSION OF THE A MATRIX.  INITIALLY IT IS
*              SET EQUAL TO SOS, BUT THEN IS UPDATED TO KEEP TRACK OF THE
*              DIMENSION OF THE A22 SUBMATRIX.
*
*  SIZCHK   :  CONSTANT CONTAINING THE PRODUCT OF NORM AND THE MACHINE
```

```
*             EPSILON.  ELEMENTS SMALLER THAN THIS IN THE SSVDC ROUTINE
*             ARE TREATED AS BEING ZERO.
*
*  SOS      :  CONSTANT CONTAINING THE DIMENSION OF THE STATE (SIZE OF
*             STATE).
*
*  STEP     :  KEEPS TRACK OF THE NUMBER OF MODIFIED OR CHAINED STEPS
*             PERFORMED.
*
*  T        :  ARRAY CONTAINING THE TRANSFORMATION MATRIX WHICH IS
*             CONTINUALLY UPDATED DURING THE ALGORITHM.
*
*  TA12     :  TEMPORARY STORAGE ARRAY WHICH IS USED DURING THE
*             COMPUTATION OF A12, SINCE A12 IS COMPUTED IN TWO STAGES.
*
*  TDMB     :  TEMPORARY STORAGE ARRAY WHICH IS USED DURING THE UPDATING
*             OF THE TRANFORMATION MATRTIX, BECAUSE THE UPDATE MUST BE
*             DONE IN TWO STAGES.
*
*  U        :  ORTHOGONAL MATRIX WHOSE COLUMNS CONTAIN THE LEFT SINGULAR
*             VECTORS OF THE MATRIX PASSED TO THE SSVDC ROUTINE.
*
*  V        :  ORTHOGONAL MATRIX WHOSE COLUMNS ARE THE RIGHT SINGULAR
*             VECTORS OF THE MATRIX PASSED TO THE SSVDC ROUTINE.
*
*  WORK     :  VECTOR USED IN THE SSVDC ROUTINE.
*
*********************************************************************
*
*     INITIALIZE THE COUNTING VARIABLES
      NOZERO = 0
      NIDE   = 0
      NM     = 100
      NZB    = 0
      STEP   = 1
*
*     FREQUENTLY USED FORMAT STATEMENTS
10    FORMAT (A)
50    FORMAT (100(F12.5))
*
      WRITE  (6,100)
100   FORMAT (/'*****************************************'
              /'**  MODIFIED OR CHAINED AGGREGATION  **'
              /'*****************************************')
*
*********************************
*  INPUT THE SYSTEM MATRICES  *
*********************************
*
      NAME  = '"C" MATRIX.'
      CALL  INPUT(C,NRC,NCC,NAME,CFRMT,FCNAME)
*
*     DETERMINE THE SMALLEST OF THE TWO DIMENSIONS
```

```
      NMIN   = MIN(NRC,NCC)
*
*     INITIALIZE THE SIZE OF THE SYSTEM AND THE SIZE OF A
      SOS    = NCC
      SIZA   = SOS
*
      NAME  = '"A" MATRIX.'
      CALL  INPUT(A,SOS,SOS,NAME,AFRMT,FANAME)
*
*     SELECT THE NORM TO BE USED FOR THE A MATRIX.
625   WRITE  (6,650)
650   FORMAT (/'ENTER THE NORM OF THE SYSTEM MATRIX, A. '
              /'  1.  LARGEST ELEMENT OF A '
              /'  2.  TWO-NORM (LARGEST SINGULAR VALUE)'
              /'  3.  FROBENIUS NORM'
              /'      ENTER 1, 2, OR 3 >',$)
*
      READ   (5,*)  I
      IF     (I .EQ. 3) GO TO 800
      IF     (I .EQ. 2) GO TO 750
      IF     (I .NE. 1) GO TO 625
*
*  FIND THE LARGEST ELEMENT OF A AND USE THIS AS THE NORM.
*
      NORM   =  0.0
      DO  700   J=1, SOS
        DO  700   I=1, SOS
          IF     (ABS(A(I,J)) .GE. NORM)  NORM = ABS(A(I,J))
700   CONTINUE
      GO TO 850
*
*  COMPUTE THE TWO NORM
*
      CALL   DUP(A,DMB,SOS,SOS)
750   CALL   SSVDC(DMB,NM,SOS,SOS,SIGMA,E,U,NM,V,NM,WORK,00,IERR)
      IF     (IERR .EQ. 0)  GO TO 760
      WRITE  (6,755)
755   FORMAT (/'SINGULAR VALUES OF A NOT COMPUTED CORRECTLY'
              /'SEE LINPACK MANUAL FOR CASE WHEN (IERR .NE. 0)')
      STOP
*
760   NORM   =  SIGMA(1)
      GO TO 850
*
*  COMPUTE THE FROBENIUS NORM
*
800   NORM   =  0
      DO  825   I=1, SOS
        DO  825  J=1, SOS
          NORM  =  NORM  +  A(I,J)*A(I,J)
825   CONTINUE
      NORM   =  SQRT(NORM)
*
```

```
850   WRITE  (6,860)  NORM
860   FORMAT (/'THE NORM OF A EQUALS: ',F12.3)
*
*     CALCULATE SIZCHK: ELEMENTS SMALLER THAN SIZCHK
*     ARE TREATED AS ZEROES.  (MACHINE EPSILON = 2.0**(-24))
*
      SIZCHK = ABS(NORM*((2.0)**(-24)))
      NAME  = '"B" MATRIX.'
      CALL  INPUT(B,NRB,NCB,NAME,BFRMT,FBNAME)
*
900   CONTINUE
*****************************************************
*  CHOOSE BETWEEN PERFORMING MODIFIED OR CHAINED  *
*****************************************************
*
      FCHND   = 0
950   WRITE   (6,1000)
1000  FORMAT  (/'DO YOU WANT TO PERFORM MODIFIED'
               /'OR CHAINED AGGREGATION?'
               /'     1) MODIFIED'
               /'     2) CHAINED'
               /'ENTER 1 OR 2  >',$)
*
      READ    (5,10) ANS
      IF      (ANS .EQ. '1') GO TO 2200
      IF      (ANS .EQ. '2') GO TO 2000
*
      WRITE   (6,1100)
1100  FORMAT  (/'INCORRECT ENTRY.',//)
      GO TO 950
*
2000  CONTINUE
*  SET THE CHAINED FLAG
      FCHND   = 1
*
*  STORE THE NUMBER OF COLUMNS IN THE ORIGINAL B
*  IN A TEMPORARY NAME
      NCBTMP  = NCB
      NCB     = 1
*
*  CREATE THE NEEDED NULL B MATRIX FOR CHAINED
      DO 2100   I=1,NRB
         B(I,1) = 0.0
2100  CONTINUE
2200  CONTINUE
*****************************************
*  COMPRESS THE COLUMNS OF THE C MATRIX  *
*****************************************
*
*  EVENTUALLY WE WILL ONLY NEED TO HAVE THE V MATRIX RETURNED.
*  FOR NOW WE WANT TO HAVE U AND V AVAILABLE FOR CHECKING PURPOSES.
*
      CALL    SSVDC(C,NM,NRC,NCC,SIGMA,E,U,NM,V,NM,WORK,11,IERR)
```

```
*
*  COMPUTE THE NUMBER OF NON-ZERO SINGULAR VALUES.
*  DISPLAY THE U AND V MATRIX IF YOU WISH.
*
      NAME   = 'THE C MATRIX'
*
      CALL    USV(U,SIGMA,V,SIZCHK,NMIN,NZ,NRC,NCC,NAME)
*
*  CHECK THE SIZE OF NZ FOR THE TRIVIAL CASE.
      IF     (NZ .EQ. 0)  GO TO 60600
*
*****************************************
*  INITIALIZE THE TRANSFORMATION MATRIX  *
*****************************************
      DO 4000   I=1,SOS
         DO 4000   J=1,SOS
         T(I,J)=V(J,I)
4000  CONTINUE
      WRITE  (6,4200)
4200  FORMAT (/'WOULD YOU LIKE TO SEE THE T MATRIX?',$)
      READ   (5,10)   ANS
      IF     ((ANS .NE. 'Y').AND.(ANS .NE. 'y'))   GO TO 4500
      CALL  OUTPUT(T,SOS,SOS,SIZCHK)
4500  CONTINUE
*
*  CHECK IF THE "C" MATRIX HAS RANK EQUAL TO THE ORDER OF THE SYSTEM.
*
      IF     (NZ .EQ. SOS)  GO TO 60700
*
***********************************
*  CALCULATE THE A12 SUBMATRIX  *
***********************************
*
*  SET THE A12 FLAG EQUAL TO ZERO
      FA12   = 0
      DO 5000   I=1,NZ
         DO 5000   J=1,SIZA
            TA12(I,J)  = 0.0
            DO 5000   K=1,SIZA
               TA12(I,J)  = TA12(I,J) + V(K,I) * A(K,J)
5000  CONTINUE
      DO 5500   I=1,NZ
         DO 5500   J=1,SIZA - NZ
            A12(I,J)  = 0.0
            JD  = NZ + J
            DO 5500   K=1,SIZA
               A12(I,J)  = A12(I,J) + TA12(I,K) * V(K,JD)
               IF   (ABS(A12(I,J)) .GE. SIZCHK)   FA12 = 1
5500  CONTINUE
*
      WRITE  (6,5600)
5600  FORMAT (/'WOULD YOU LIKE TO SEE THE A12 SUBMATRIX? >',$)
      READ   (5,10)  ANS
```

```
      IF     ((ANS .NE. 'Y').AND.(ANS .NE. 'y'))  GO TO 5850
      CALL  OUTPUT(A12,NZ,SIZA-NZ,SIZCHK)
*
5850  CONTINUE
*  WAS A12 = 0  (FA12 = 0)
      IF   (FA12 .EQ. 0)   GO TO 50000
*
5900  CONTINUE
*
*******************************
*  COMPUTE THE NEW B MATRIX   *
*******************************
*
      DO 6000   I=1,SIZA
         DO 6000   J=1,NCB
            G(I,J) = 0.0
            DO 6000   K=1,SIZA
               G(I,J) = G(I,J) + V(K,I) * B(NIDE+K,J)
6000  CONTINUE
*
      IF     (FCHND .EQ. 1) GO TO 6450
*
      WRITE  (6,6050)
6050  FORMAT (/'WOULD YOU LIKE TO SEE THE NEW PORTION OF B? >',$)
      READ   (5,10)  ANS
      IF     ((ANS .NE. 'Y').AND.(ANS .NE. 'y'))  GO TO 6150
      CALL  OUTPUT(G,SIZA,NCB,SIZCHK)
*
6150  CONTINUE
*  STORE THIS UPDATED SECTION OF THE B MATRIX
*  IN THE APPROPRIATE LOCATION OF THE ENTIRE B MATRIX.
*
      DO 6200   I=1,SIZA
         ID = NIDE + I
         DO 6200   J=1,NCB
            B(ID,J) = G(I,J)
6200  CONTINUE
*
      WRITE  (6,6300)
6300  FORMAT (/'WOULD YOU LIKE TO SEE THE ENTIRE B MATRIX? >',$)
      READ   (5,10)  ANS
      IF     ((ANS .NE. 'Y').AND.(ANS .NE. 'y'))  GO TO 6450
      CALL  OUTPUT(B,SOS,NCB,SIZCHK)
*
6450  CONTINUE
*  CREATE A DUMMY MATRIX COMPOSED OF
*  THE FIRST NZB+NZ ROWS OF B, STARTING
*  WITH THE FIRST NON-ZERO ROW (NOZERO+1).
*
      DO 6500   I=1,NZB+NZ
         ID = NOZERO + I
         DO 6500   J=1,NCB
            GDMB(I,J) = B(ID,J)
```

```
6500  CONTINUE
*
      IF      (FCHND .EQ. 1) GO TO 6750
*
      WRITE   (6,6600)
6600  FORMAT (/'WOULD YOU LIKE TO SEE THE SUBMATRIX'
              /'OF B WHICH IS TO BE DECOMPOSED? >',$)
      READ    (5,10)  ANS
      IF      ((ANS .NE. 'Y').AND.(ANS .NE. 'y'))  GO TO 6750
      CALL   OUTPUT(GDMB,NZB+NZ,NCB,SIZCHK)
*
6750  CONTINUE
      NRGDMB = NZB + NZ
      NMINB  = MIN(NRGDMB,NCB)
*
      CALL      DUP(GDMB,DMB,NRGDMB,NCB)
*
*  PASS THE DMB TO SSVDC TO COMPRESS ITS ROWS.
*  ONLY RETURN THE U MATRIX.  WE DO NOT WANT TO OVER-
*  WRITE THE CURRENT V MATRIX.
*
      CALL    SSVDC(DMB,NM,NRGDMB,NCB,SIGMA,E,U,NM,V,NM,WORK,10,IERR)
*
      IF      (FCHND .EQ. 0) GO TO 6800
      NZB     = 0
*  SKIP B AND T UPDATE
      GO TO 10465
*
6800  CONTINUE
*  COMPUTE THE NUMBER OF NON-ZERO SINGULAR VALUES.
*  DISPLAY THE U MATRIX IF DESIRED.
*
      NAME    = 'A B SUBMATRIX.'
*
      CALL    USV(U,SIGMA,V,SIZCHK,NMINB,NZB,NRGDMB,NCB,NAME)
*
*  CHECK TO SEE IF GDMB WAS EQUAL TO ZERO.
*  IF SO, THEN SKIP THE B AND T UPDATE.
*
      IF      (NZB .EQ. 0)  GO TO 10465
*
*************************
*  UPDATE THE B MATRIX  *
*************************
*
*  ZERO OUT THE APPROPRIATE ELEMENTS IN THE B MATRIX.
*  THIS SUBMATRIX IS THE SIZE OF GDMB.
*
      DO 10000 I=1,NRGDMB
         ID = NOZERO + I
         DO 10000 J=1,NCB
            B(ID,J) = 0.0
10000 CONTINUE
```

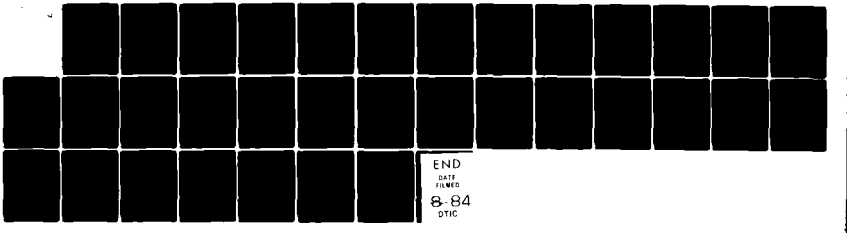
```
*
*  PERFORM THE TRANSFORMATION ON THE B MATRIX
*  WITH THE U MATRIX, COMPRESSING THE ROWS OF
*  THE NEW B DOWN.
*
      DO 10100   I=1,NZB
         ID = NOZERO + NRGDMB - NZB + I
         DO 10100   J=1,NCB
            DO 10100   K=1,NRGDMB
               B(ID,J) = B(ID,J) + U(K,I) * GDMB(K,J)
10100 CONTINUE
*
      WRITE  (6,10150)
10150 FORMAT (/'WOULD YOU LIKE TO SEE THE TRANSFORMED B MATRIX'
              /'AFTER THE APPROPRIATE ROWS HAVE BEEN COMPRESSED? >',$)
      READ   (5,10)  ANS
      IF     ((ANS .NE. 'Y').AND.(ANS .NE. 'y'))  GO TO 10165
      CALL  OUTPUT(B,SOS,NCB,SIZCHK)
*
10165 CONTINUE
****************************************
*  UPDATE THE TRANSFORMATION MATRIX  *
****************************************
*
*  THIS MUST BE DONE IN TWO STAGES, BECAUSE WE
*  COMPRESS THE ROWS DOWN IN THE B SUBMATRIX.
*
*  CALCULATE THE UPPER PART OF THE NEW T MATRIX.
*
      DO 10210   I=1,NRGDMB-NZB
         ID = NZB + I
         DO 10210   J=1,SOS
            TDMB(I,J) = 0.0
            DO 10210   K=1,NRGDMB
               TDMB(I,J) = TDMB(I,J) + U(K,ID) * T(NOZERO+K,J)
10210 CONTINUE
*
*  CALCULATE THE LOWER PART OF THE NEW T MATRIX.
*
      DO 10220   I=1,NZB
         ID = NRGDMB - NZB + I
         DO 10220   J=1,SOS
            TDMB(ID,J) = 0.0
            DO 10220   K=1,NRGDMB
               TDMB(ID,J) = TDMB(ID,J) + U(K,I) * T(K+NOZERO,J)
10220 CONTINUE
*
      WRITE  (6,10250)
10250 FORMAT (/'WOULD YOU LIKE TO SEE THE UPDATED ROWS IN T'
              /'DUE TO THE U GENERATED WHILE COMPRESSING THE'
              /'ROWS OF B? >',$)
      READ   (5,10)  ANS
      IF     ((ANS .NE. 'Y').AND.(ANS .NE. 'y'))  GO TO 10265
```

```
      CALL  OUTPUT(TDMB,NRGDMB,SOS,SIZCHK)
*
10265 CONTINUE
*
*  STORE TDMB IN THE APPROPRIATE LOCATION WITHIN T
*
      DO 10300   I=1,NRGDMB
         ID = NOZERO + I
         DO 10300   J=1,SOS
            T(ID,J) = TDMB(I,J)
10300 CONTINUE
*
      WRITE  (6,10350)
10350 FORMAT (/'WOULD YOU LIKE TO SEE THE NEW T? >',$)
      READ   (5,10)  ANS
      IF     ((ANS .NE. 'Y').AND.(ANS .NE. 'y'))  GO TO 10365
      CALL  OUTPUT(T,SOS,SOS,SIZCHK)
*
10365 CONTINUE
*
*****************************************
*  UPDATE THE A12 SUBMATRIX WITH THIS U  *
*  STATE SPACE TRANSFORMATION.           *
*****************************************
*
*  A12 WILL BE DIVIDED INTO TWO PARTS, A12B AND A12S.
*  CALCULATE A12B.
      DO 10400   I=1,NZB
         DO 10400   J=1,SIZA-NZ
            A12B(I,J) = 0.0
            DO 10400   K=1,NRGDMB
               A12B(I,J) = A12B(I,J) + U(K,I) * A12(K,J)
10400 CONTINUE
*
      WRITE  (6,10450)
10450 FORMAT (/'WOULD YOU LIKE TO SEE THE A12B SUBMATRIX? >',$)
      READ   (5,10)  ANS
      IF     ((ANS .NE. 'Y').AND.(ANS .NE. 'y'))  GO TO 10465
      CALL  OUTPUT(A12B,NZB,SIZA-NZ,SIZCHK)
*
10465 CONTINUE
*
*  CALCULATE A12S.
*  SET THE A12S=0 FLAG EQUAL TO ZERO.
      FA12S  = 0
      DO 10500   I=1,NRGDMB - NZB
         ID = NZB + I
         DO 10500   J=1,SIZA-NZ
            A12S(I,J) = 0.0
            DO 10500   K=1,NRGDMB
               A12S(I,J) = A12S(I,J) + U(K,ID) * A12(K,J)
               IF  (ABS(A12S(I,J)) .GT. SIZCHK)   FA12S = 1
10500 CONTINUE
```

```
*
      WRITE  (6,10550)
10550 FORMAT (/'WOULD YOU LIKE TO SEE THE A12S SUBMATRIX? >',$)
      READ   (5,10)  ANS
      IF     ((ANS .NE. 'Y').AND.(ANS .NE. 'y'))  GO TO 10565
      CALL  OUTPUT(A12S,NRGDMB-NZB,SIZA-NZ,SIZCHK)
*
10565 CONTINUE
*
      IF     (FA12S .EQ. 0)   GO TO 50200
*
**********************************
*  CALCULATE THE A22 SUBMATRIX.  *
**********************************
*
*  TA12 IS USED AS THE INTERMEDIATE STORAGE LOCATION.
*
      DO 10600   I=1,SIZA-NZ
         ID = NZ + I
         DO 10600   J=1,SIZA
            TA12(I,J) = 0.0
            DO 10600   K=1,SIZA
               TA12(I,J) = TA12(I,J) + V(K,ID) * A(K,J)
10600 CONTINUE
*
*  STORE THE A22 SUBMATRIX IN THE UPPER LEFT HAND
*  CORNER OF THE A MATRIX.
*
      DO 10700   I=1,SIZA-NZ
         DO 10700   J=1,SIZA-NZ
            JD = NZ + J
            A(I,J) = 0.0
            DO 10700   K=1,SIZA
               A(I,J) = A(I,J) + TA12(I,K) * V(K,JD)
10700 CONTINUE
*
      WRITE  (6,10750)
10750 FORMAT (/'WOULD YOU LIKE TO SEE THE A22 SUBMATRIX?  >',$)
      READ   (5,10)  ANS
      IF     ((ANS .NE  'Y').AND.(ANS .NE. 'y'))  GO TO 10760
      CALL  OUTPUT(A,SIZA-NZ,SIZA-NZ,SIZCHK)
10760 CONTINUE
*
*  UPDATE THE SIZA VARIABLE TO BE THE SIZE OF THE
*  NEW A22 SUBMATRIX.
*
      SIZA = SIZA - NZ
*
*  UPDATE THE VARIABLE REPRESENTING THE NUMBER OF
*  IDENTITY ELEMENTS.  (THIS VARIABLE IS USED DURING
*  THE T UPDATE WITH V.)
*
      NIDE = NIDE + NZ
```

AD-A142 394 A NUMERICAL ALGORITHM FOR CHAINED AGGREGATION AND MODIFIED CHAINED AGGREGATION(U) ILLINOIS UNIV AT URBANA DECISION AND CONTROL LAB H S THARP SEP 83 DC-62 2/2

UNCLASSIFIED N00014-79-C-0424 F/G 12/1 NL

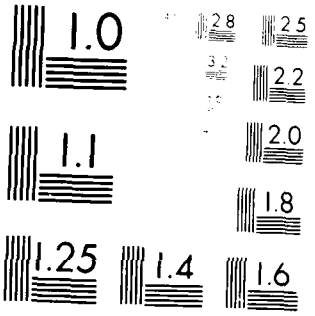


END

DATE FILMED

8-84

DTIC



```
*
**************************************
*  IDENTIFICATION OF STEP TO USER  *
**************************************
*
*  INDICATE TO USER WHAT STEP IN THE DECOMPOSITION
*  HE IS IN.
*
      NAME   =  '        '
      IF     (FCHND .EQ. 0)   NAME = 'MODIFIED'
      WRITE  (6,10770)  STEP  ,  NAME
10770 FORMAT (/'*****************************************'
              /'THIS COMPLETES ',I3,'  STAGE(S) OF ',A8,
              /'CHAINED AGGREGATION.')
      WRITE  (6,10772)  NIDE , NIDE
10772 FORMAT (/'THE AGGREGATE SYSTEM IS NOW ',I3,'   x ',I3,
              /'*****************************************')
      IF     (STEP .NE. 1)  GO TO 10780
      WRITE  (6,10774)
10774 FORMAT (/'WOULD YOU LIKE THIS INFORMATION TO BE'
              /'WRITTEN OUT TO A FILE?  >',$)
      READ   (5,10)  ANS
      IF     ((ANS .EQ. 'Y').OR.(ANS .EQ. 'y'))  GO TO 10775
      INFO   = .FALSE.
      GO TO 10780
10775 INFO   = .TRUE.
      WRITE  (6,10776)
10776 FORMAT (/'ENTER THE FILE NAME  >',$)
      READ   (5,10)  INAME
      OPEN   (UNIT=3,FILE=INAME)
      REWIND 3
      WRITE  (3,10778)
10778 FORMAT (/'THIS FILE CONTAINS THE INFORMATION ON'
              /'HOW THE AGGREGATE SYSTEM GROWS WITH'
              /'EACH STEP OF MODIFIED CHAINED AGGREGATION.',/)
      CLOSE  (UNIT=3)
*
10780 IF     (.NOT. INFO)  GO TO 10790
      OPEN   (UNIT=3,FILE=INAME)
      WRITE  (3,10785)  STEP , NIDE , NIDE
10785 FORMAT  (/'AFTER ',I3,'  STAGE(S) OF MODIFIED CHAINED'
               /'AGGREGATION, THE AGGREGATE SYSTEM IS'
               /,I3,'   x ',I3)
*
10790 CONTINUE
      STEP   = STEP + 1
*
*
**************************************
*  COMPRESS THE COLUMNS OF A12S  *
**************************************
*
*  WILL RETURN BOTH U AND V MATRICES FOR NOW.
```

```
*
      NRA12S = NRGDMB - NZB
*  RECALL NCA12S = SIZA.
*
      CALL  SSVDC(A12S,NM,NRA12S,SIZA,SIGMA,E,U,NM,V,NM,WORK,11,IERR)
*
      NAME = 'THE A12S SUBMATRIX.'
      NMIN = MIN(NRA12S,SIZA)
*
      CALL  USV(U,SIGMA,V,SIZCHK,NMIN,NZ,NRA12S,SIZA,NAME)
*
*  IF NZ EQUALS SIZA THEN THE SYSTEM WILL NOT AGGREGATE.
*  THIS IS BECAUSE A12S HAS FULL COLUMN RANK.
*
      IF  (NZ .EQ. SIZA)  GO TO 60800
*
**************
*  UPDATE T  *
**************
*
*  DO THE UPDATE IN TWO STEPS.
*  COMPUTE AFFECTED ROWS OF T.
*
      DO 10800   I=1,SIZA
         DO 10800   J=1,SOS
            TDMB(I,J) = 0.0
            DO 10800   K=1,SIZA
               TDMB(I,J) = TDMB(I,J) + V(K,I) * T(NIDE+K,J)
10800 CONTINUE
*
*  STORE THESE AFFECTED ROWS IN THE APPROPRIATE LOCATION
*  WITHIN T.
*
      DO 10900   I=1,SIZA
         ID = NIDE + I
         DO 10900   J=1,SOS
            T(ID,J) = TDMB(I,J)
10900 CONTINUE
*
      WRITE  (6,10950)
10950 FORMAT (/'WOULD YOU LIKE TO SEE THIS NEW T'
              /'AFTER BEING UPDATED WITH V?  >',$)
      READ   (5,10)  ANS
      IF     ((ANS .NE. 'Y').AND.(ANS .NE. 'y')) GO TO 10970
      CALL  OUTPUT(T,SOS,SOS,SIZCHK)
10970 CONTINUE
*
************************************
*  COMPUTE THE NEW A12 SUBMATRIX  *
************************************
*
*  SET THE A12=0 FLAG TO ZERO.
      FA12 = 0
```

```
*
*  CALCULATE THE UPPER HALF OF THE A12 SUBMATRIX.
*
      DO 11000   I=1,NZB
         DO 11000   J=1,SIZA-NZ
            JD = NZ + J
            A12(I,J) = 0.0
            DO 11000   K=1,SIZA
               A12(I,J) = A12(I,J) + A12B(I,K) * V(K,JD)
               IF  (ABS(A12(I,J)) .GT. SIZCHK)  FA12 = 1
11000 CONTINUE
*
*  CALCULATE THE LOWER HALF OF THE A12 SUBMATRIX IN
*  TWO SECTIONS.
*
      DO 11100   I=1,NZ
         DO 11100   J=1,SIZA
            TA12(I,J) = 0.0
            DO 11100   K=1,SIZA
               TA12(I,J) = TA12(I,J) + V(K,I) * A(K,J)
11100 CONTINUE
*
      DO 11200   I=1,NZ
         ID = NZB + I
         DO 11200   J=1,SIZA - NZ
            JD = NZ + J
            A12(ID,J) = 0.0
            DO 11200   K=1,SIZA
               A12(ID,J) = A12(ID,J) + TA12(I,K) * V(K,JD)
               IF  (ABS(A12(ID,J)) .GT. SIZCHK)  FA12 = 1
11200 CONTINUE
*
      WRITE  (6,11250)
11250 FORMAT (/'WOULD YOU LIKE TO SEE THE NEW A12 SUBMATRIX?  >',$)
      READ   (5,10)  ANS
      IF     ((ANS .NE. 'Y').AND.(ANS .NE. 'y'))  GO TO 11270
      CALL  OUTPUT(A12,NZB+NZ,SIZA-NZ,SIZCHK)
11270 CONTINUE
*
*  CHECK TO SEE IF A12=0.  IF SO, SYSTEM AGGREGATES.
      IF     (FA12 .EQ. 0)  GO TO 50000
*
*
*  UPDATE THE VARIABLE REPRESENTING THE NUMBER OF ZERO
*  ROWS IN THE B MATRIX.
*
      NOZERO  = NIDE - NZB
*
*****************************************************************
*  CONTINUE THE PROCESS OF MODIFIED CHAINED AGGREGATION.  *
*****************************************************************
*
40900 GO TO 5900
```

```
*
*********************
*  PROGRAM EXITS    *
*********************
*
50000 WRITE  (6,50100)
50100 FORMAT (/'PROGRAM EXIT.  SYSTEM AGGREGATES.'
              /'THE A12 SUBMATRIX = 0.')
      GO TO  70000
*
50200 WRITE  (6,50250)
50250 FORMAT (/'PROGRAM EXIT.  SYSTEM WILL AGGREGATE.'
              /'THE MATRIX A12S = 0.')
      GO TO 70000
*
*
*******************************************************
*  PROGRAM EXITS ASSOCIATED WITH NO AGGREGATION.  *
*******************************************************
*
60600 WRITE  (6,60650)
60650 FORMAT (/'PROGRAM EXIT.  TRIVIAL CASE'
              /'THE C MATRIX IS ZERO.')
      GO TO 80000
*
60700 WRITE  (6,60750)
60750 FORMAT (/'PROGRAM EXIT.  SYSTEM WILL NOT AGGREGATE.'
              /'THE C MATRIX HAS A RANK EQUAL TO THE'
              /'DIMENSION OF THE SYSTEM.')
      GO TO 70000
*
60800 WRITE  (6,60850)
60850 FORMAT (/'PROGRAM EXIT.  SYSTEM WILL NOT AGGREGATE.'
              /'THE A12S SUBMATRIX HAS FULL COLUMN RANK.')
      GO TO 70000
*
70000 CONTINUE
      WRITE  (6,70100)
70100 FORMAT (/'WOULD YOU LIKE TO SEE THE TRANSFORMED'
              /'SYSTEM MATRICES? >',$)
      READ   (5,10)  ANS
      IF     ((ANS .NE. 'Y').AND.(ANS .NE. 'y'))  GO TO 80000
*
*  COMPUTE THE TRANSFORMED A MATRIX.
*
      CALL  MULT(T,0,AFRMT,FANAME,SOS,SOS,A,1,AFRMT,FANAME,SOS,SOS,TDMB)
      DO 70200   I=1,SOS
         DO 70200   J=1,SOS
            DMB(I,J) = T(J,I)
70200 CONTINUE
*
      CALL  MULT(TDMB,0,AFRMT,FANAME,SOS,SOS,DMB,0,AFRMT,FANAME,SOS,
                 SOS,A)
```

```
*
      WRITE  (6,70300)
70300 FORMAT (/'THIS IS THE TRANSFORMED "A" MATRIX:')
      CALL OUTPUT(A,SOS,SOS,SIZCHK)
*
*  COMPUTE THE TRANSFORMED "B" MATRIX.
*
      IF     (FCHND .EQ. 1)   NCB = NCBTMP
*
      CALL  MULT(T,0,BFRMT,FBNAME,SOS,SOS,B,1,BFRMT,FBNAME,SOS,NCB,G)
*
      WRITE  (6,70400)
70400 FORMAT (/'THIS IS THE TRANSFORMED "B" MATRIX.')
      CALL  OUTPUT(G,SOS,NCB,SIZCHK)
*
*  COMPUTE THE TRANSFORMED "C" MATRIX.
*  (RECALL DMB IS THE TRANSPOSE OF THE "T" MATRIX.)
*
      CALL  MULT(C,1,CFRMT,FCNAME,NRC,NCC,DMB,0,CFRMT,FCNAME,
                 SOS,SOS,TDMB)
*
      WRITE  (6,70500)
70500 FORMAT (/'THIS IS THE TRANSFORMED "C" MATRIX.')
      CALL  OUTPUT(TDMB,NRC,NCC,SIZCHK)
*
      WRITE  (6,70600)
70600 FORMAT (/'WOULD YOU LIKE TO SEE THE FINAL "T" MATRIX? >',$)
      READ   (5,10)  ANS
      IF     ((ANS .NE. 'Y').AND.(ANS .NE. 'y'))  GO TO 70800
*
      CALL  OUTPUT(T,SOS,SOS,SIZCHK)
*
70800 WRITE  (6,70900)
70900 FORMAT (/'WOULD YOU LIKE TO SEE THE FINAL "T" TRANSPOSE'
              /'MATRIX?  >',$)
      READ   (5,10)  ANS
      IF     ((ANS .NE. 'Y').AND.(ANS .NE. 'y'))  GO TO 80000
*
      CALL  OUTPUT(DMB,SOS,SOS,SIZCHK)
*
80000 CONTINUE
      STOP
      END
*
*****************************************
*  SUBROUTINES USED IN PROGRAM ABOVE  *
*****************************************
*
**********************
*  SUBROUTINE USV  *
**********************
*
*  THIS SUBROUTINE COMPUTES THE NUMBER OF NON-ZERO
```

```
*  SINGULAR VALUES ASSOCIATED WITH A DECOMPOSED
*  MATRIX.  IT ALSO ASKS IF YOU WOULD LIKE TO SEE
*  THE U AND/OR V MATRIX CREATED DURING THE DECOMPOSITION,
*  DEPENDING ON WHERE IN THE PROGRAM USV IS CALLED.
*
      SUBROUTINE USV(U,SIGMA,V,SIZCHK,MIN,N,NR,NC,TYPE)
      CHARACTER*1       ANS
      CHARACTER*20      TYPE
      REAL          U(100,100)  , V(100,100)  , SIGMA(100)
      REAL          SIZCHK
      INTEGER       MIN         , N           , NR          , NC
      INTEGER       I           , IERR
*
*  FORMAT STATEMENTS USED
910   FORMAT (A)
950   FORMAT (100(F12.5))
*
      WRITE  (6,990)  TYPE
990   FORMAT (/'THIS DECOMPOSITION IS OF ',A)
*
      WRITE  (6,1000)
1000  FORMAT (/'WOULD YOU LIKE TO SEE THE U MATRIX?  >',$)
      READ   (5,910)  ANS
      IF     ((ANS .NE. 'Y').AND.(ANS .NE. 'y'))   GO TO 2050
*
      WRITE  (6,1500)
1500  FORMAT (/'THE CORRESPONDING U MATRIX IS:')
      CALL   DSPLAY(U,NR,NR)
*
2050  CONTINUE
*
      WRITE  (6,2100)
2100  FORMAT (/'THE SINGULAR VALUES ARE:')
      WRITE  (6,950)  (SIGMA(I),I=1,MIN)
*
*     CHECK THE MAGNITUDE OF THE SINGULAR VALUES
      N=0
      DO 2200   I=1,MIN
      IF     (SIGMA(I).LT.SIZCHK) GO TO 2300
      N=N+1
2200  CONTINUE
*
2300  WRITE  (6,2350)  N
2350  FORMAT (/'THE NUMBER OF INDEPENDENT COLUMNS IN'
              /'THE DECOMPOSED MATRIX IS:  ',I5)
*
      IF     (TYPE .EQ. 'A B SUBMATRIX.')  GO TO 3050
      WRITE  (6,2400)
2400  FORMAT (/'WOULD YOU LIKE TO SEE THE V MATRIX?  >',$)
      READ   (5,910)   ANS
      IF     ((ANS .NE. 'Y').AND.(ANS .NE. 'y'))  GO TO 3050
*
      WRITE  (6,2500)
```

```
2500  FORMAT (//'THE CORRESPONDING V MATRIX IS:')
      CALL   DSPLAY(V,NC,NC)
*
3050  CONTINUE
*
      WRITE  (6,3500) IERR
3500  FORMAT (//'IERR EQUALS ',I4)
      RETURN
      END
*
**********************
*  SUBROUTINE DUP  *
**********************
*
*  THIS SUBROUTINE DUPLICATES A MATRIX.  THIS IS NEEDED
*  SINCE WHEN A MATRIX IS PASSED TO SSVDC IT RETURNS IN
*  AN ALTERED FORM.
*
      SUBROUTINE DUP(SOURCE,DUMMY,NROW,NCOL)
*
      REAL         SOURCE(100,100)     , DUMMY(100,100)
      INTEGER      NROW                , NCOL
      INTEGER      I                   , J
*
      DO 1000   I=1,NROW
         DO 1000   J=1,NCOL
            DUMMY(I,J) = SOURCE(I,J)
1000  CONTINUE
      RETURN
      END
*
************************
*  SUBROUTINE OUTPUT  *
************************
*
*  THIS SUBROUTINE ALLOWS THE USER TO SEE THE
*  PARTICULAR MATRIX ON THE SCREEN IN EITHER F6.3
*  FORMAT OR SIMPLY AS X's AND O's (IF THE STRUCTURE
*  IS ALL THAT IS DESIRED).  THE USER MAY ALSO OUTPUT
*  THE MATRIX TO A FILE IN EITHER OF THESE FORMATS.
*  THE USER MUST SUPPLY THE FILENAME TO BE USED.
*
      SUBROUTINE OUTPUT(X,NROW,NCOL,SIZCHK)
      CHARACTER*1         ANS
      CHARACTER*1         XC(100,100)
      CHARACTER*20        FNAME
      REAL                X(100,100)     , SIZCHK
      INTEGER             I              , J
      INTEGER             NROW           , NCOL
      INTEGER             FORM
      LOGICAL             FOUT
*
*  FREQUENTLY USED FORMAT STATEMENTS.
```

```
*
10    FORMAT (8(F9.3,1X))
15    FORMAT (7(E15.7,1X))
20    FORMAT (A)
30    FORMAT (40(A,2X))
*
50    WRITE  (6,100)
100   FORMAT (/'WHAT FORMAT DO YOU WISH TO SEE'
              /'THIS MATRIX IN?'
              /'     1)  NUMERICAL VALUES'
              /'     2)  STRUCTURE ONLY (X AND O)'
              /'ENTER 1 OR 2  >',$)
*
      READ   (5,150) FORM
150   FORMAT (I1)
*
      WRITE  (6,200)
200   FORMAT (/'WOULD YOU LIKE THIS MATRIX STORED'
              /'IN A FILE?  >',$)
*
      READ   (5,20)  ANS
      IF     ((ANS .EQ. 'Y').OR.(ANS .EQ. 'y'))  GO TO 275
      FOUT = .FALSE.
      GO TO 400
*
275   WRITE  (6,300)
300   FORMAT (/'ENTER THE FILENAME FOR THIS MATRIX >',$)
      READ   (5,20)  FNAME
      FOUT = .TRUE.
*
400   IF     (FORM .EQ. 1)  GO TO 500
      IF     (FORM .EQ. 2)  GO TO 1000
      WRITE  (6,450)
450   FORMAT (/'INCORRECT ENTRY(S).',//)
      GO TO 50
*
500   CALL    DSPLAY(X,NROW,NCOL)
      IF     (.NOT. FOUT)  RETURN
*
600   OPEN   (UNIT=1,FILE=FNAME)
      REWIND 1
      WRITE  (1,*)  NROW , NCOL
      DO 650   I=1,NROW
         WRITE  (1,15)  (X(I,J),J=1,NCOL)
650   CONTINUE
      CLOSE  (UNIT=1)
      RETURN
*
*
1000  DO 1050   I=1,NROW
         DO 1050   J=1,NCOL
            XC(I,J) = 'O'
            IF  (ABS(X(I,J)) .GT. SIZCHK)  XC(I,J) = 'X'
```

```
1050  CONTINUE
*
      IF     (FOUT)  GO TO 1300
      DO 1100   I=1,NROW
         WRITE  (6,30)  (XC(I,J),J=1,NCOL)
1100  CONTINUE
      RETURN
*
1300  OPEN   (UNIT=1,FILE=FNAME)
      REWIND 1
      DO 1400   I=1,NROW
         WRITE  (6,30)  (XC(I,J),J=1,NCOL)
         WRITE  (1,30)  (XC(I,J),J=1,NCOL)
1400  CONTINUE
      CLOSE  (UNIT=1)
      RETURN
      END
*
**********************
*  SUBROUTINE INPUT  *
**********************
*
*  THIS SUBROUTINE READS IN THE APPROPRIATE MATRIX FROM
*  A DATA FILE.  THE MATRIX MAY BE STORED IN NORMAL FORM
*  OR AS A SPARSE MATRIX.  YOU ARE ALSO ALLOWED THE ABILITY
*  TO VERIFY THE MATRIX READ IN.
*
      SUBROUTINE INPUT(X,NROW,NCOL,TYPE,FRMT,FNAME)
      CHARACTER*1       ANS         , FRMT
      CHARACTER*20      FNAME       , TYPE
      REAL              X(100,100)
      INTEGER           I           , J
      INTEGER           NROW        , NCOL
      INTEGER           NR          , NC
*
      WRITE  (6,200)  TYPE
200   FORMAT (/'ENTER THE NAME OF THE DATA FILE FOR THE ',A11,' >',$)
      READ   (5,300)  FNAME
300   FORMAT (A)
      IF     (TYPE .EQ. '"A" MATRIX.')  GO TO 252
      WRITE  (6,250)  TYPE
250   FORMAT (/'ENTER THE DIMENSIONS OF THE ',A10,' (ROWS x COLS). >',$)
      READ   (5,*) NROW,NCOL
*
252   WRITE  (6,255)
255   FORMAT (/'WHAT FORMAT IS THIS MATRIX IN?'
              /'     A)  SPARSE'
              /'     B)  NORMAL'
              /'     ENTER A OR B  >',$)
      READ   (5,300)  FRMT
      IF     ((FRMT .EQ. 'A').OR.(FRMT .EQ. 'a'))  GO TO 310
      IF     ((FRMT .EQ. 'B').OR.(FRMT .EQ. 'b'))  GO TO 305
      GO TO 252
```

```
*
305   OPEN   (UNIT=1,FILE=FNAME)
      REWIND 1
      READ   (1,*)  NR  ,  NC
      IF     ((NR .EQ. NROW).AND.(NC .EQ. NCOL))  GO TO 307
      WRITE  (6,306)
306   FORMAT (/'PROGRAM EXIT!  MATRIX DIMENSIONS WHICH WERE'
              /'INPUT AND THOSE IN THE DATAFILE DO NOT AGREE.')
      STOP
307   READ   (1,*)   ((X(I,J),J=1,NCOL),I=1,NROW)
      CLOSE  (UNIT=1)
      GO TO 350
*
*
310   DO 308   I=1,NROW
         DO 308   J=1,NCOL
            X(I,J)=0.0
308   CONTINUE
*
      OPEN   (UNIT=1,FILE=FNAME)
      REWIND 1
312   READ   (1,*,END=350)  I , J , X(I,J)
      GO TO 312
*
350   WRITE  (6,360)  TYPE
360   FORMAT (/'DO YOU WANT TO VERIFY THE ',A10,'? >',$)
      READ   (5,300)  ANS
      IF     ((ANS.NE.'Y').AND.(ANS.NE.'y'))     RETURN
      WRITE  (6,400)  TYPE
400   FORMAT (/'THE FOLLOWING ',A10,' WAS READ IN:')
      CALL   DSPLAY(X,NROW,NCOL)
1000  CONTINUE
*
      RETURN
      END
*
**********************
*  SUBROUTINE MULT   *
**********************
*
*  THIS SUBROUTINE MULTIPLIES TWO MATRICES TOGETHER AND
*  RETURNS THE RESULT IN P.  ONE OR BOTH OF THE FILES MAY
*  HAVE TO BE READ IN FROM DATAFILES, THIS OPTION IS CON-
*  TROLLED BY A FLAG FOR EACH MATRIX.
*     1 MEANS READ THE FILE IN FROM A DATAFILE.
*     0 MEANS THE MATRIX WAS PASSED IN THE CALL.
*     *FMT INDICATES THE FORMAT THE PARTICULAR
*          MATRIX IS STORED IN.
*
      SUBROUTINE MULT(X,XF,XFMT,XNAME,NRX,NCX,Y,YF,YFMT,YNAME,NRY,
                      NCY,P)
      CHARACTER*1     XFMT     , YFMT
      CHARACTER*20    XNAME    , YNAME
```

```
      LOGICAL          XF         , YF
      INTEGER          I          , J          , K
      INTEGER          NRX        , NRY        , NCX        , NCY
      INTEGER          RD         , CD
      REAL             X(100,100), Y(100,100), P(100,100)
*
      IF     (NCX .EQ. NRY)  GO TO 50
      WRITE  (6,30)
30    FORMAT (/'THE MATRICES ARE NOT COMPATABLE!')
      STOP
*
*  DOES THE MATRIX NEED TO BE READ IN FROM A DATA FILE?
*
50    IF     (.NOT.XF)  GO TO 500
*
*  WHAT FORMAT IS THE MATRIX IN? (A=SPARSE,B=NORMAL)
*
      IF     ((XFRMT .EQ. 'A').OR.(XFRMT .EQ. 'a'))  GO TO 300
      IF     ((XFRMT .EQ. 'B').OR.(XFRMT .EQ. 'b'))  GO TO 200
      WRITE  (6,100)
100   FORMAT (/'THE X MATRIX IN THE MULT SUBROUTINE'
              /'WAS NOT ASSIGNED A FORMAT TYPE')
      STOP
*
200   OPEN   (UNIT=1,FILE=XNAME)
      REWIND 1
      READ   (1,*)  RD  ,  CD
      READ   (1,*)  ((X(I,J),J=1,NCX),I=1,NRX)
      CLOSE  (UNIT=1)
      GO TO 500
*
300   DO 350   I=1,NRX
         DO 350   J=1,NCX
            X(I,J) = 0.0
350    CONTINUE
*
      OPEN   (UNIT=1,FILE=XNAME)
      REWIND 1
360   READ   (1,*,END=500) I , J , X(I,J)
      GO TO 360
*
*  DOES THE Y MATRIX NEED TO BE READ IN FROM A DATAFILE?
*
500   IF     (.NOT.YF)  GO TO 1000
*
*  WHAT FORMAT IS THE "Y" MATRIX IN? (A=SPARSE,B=NORMAL)
*
      IF     ((YFRMT .EQ. 'A').OR.(YFRMT .EQ. 'a'))  GO TO 800
      IF     ((YFRMT .EQ. 'B').OR.(YFRMT .EQ. 'b'))  GO TO 600
      WRITE  (6,550)
550   FORMAT (/'THE "Y" MATRIX IN THE MULT SUBROUTINE'
              /'WAS NOT ASSIGNED A FORMAT TYPE')
      STOP
```

```
*
600   OPEN   (UNIT=1,FILE=YNAME)
      REWIND 1
      READ   (1,*)  RD  , CD
      READ   (1,*)  ((Y(I,J),J=1,NCY),I=1,NRY)
      CLOSE  (UNIT=1)
      GO TO 1000
*
800   DO 850   I=1,NRY
         DO 850   J=1,NCY
            Y(I,J) = 0.0
850   CONTINUE
*
      OPEN   (UNIT=1,FILE=YNAME)
      REWIND 1
860   READ   (1,*,END=1000) I , J , Y(I,J)
      GO TO 860
*
1000  DO 1200   I=1,NRX
         DO 1200   J=1,NCY
            P(I,J) = 0.0
            DO 1200   K=1,NCX
               P(I,J) = P(I,J) + X(I,K)*Y(K,J)
1200  CONTINUE
      RETURN
      END
*
************************
*  SUBROUTINE DSPLAY   *
************************
*
*  THIS SUBROUTINE ALLOWS A MATRIX TO BE
*  DISPLAYED ON THE TERMINAL SCREEN IN
*  GROUPS OF 8 COLUMNS.
*
      SUBROUTINE DSPLAY(X,NROW,NCOL)
      INTEGER     I         , J         , START
      INTEGER     NROW      , NCOL      , FINISH
      REAL        X(100,100)
*
      START  =  1
      FINISH =  8
100   IF     (FINISH .GE. NCOL)  FINISH = NCOL
      WRITE  (6,150)  START , FINISH
150   FORMAT (/' COLS. ',I3,' TO ',I3)
      DO 200  I=1, NROW
        WRITE  (6,175)  (X(I,J),J=START,FINISH)
175   FORMAT (8(F9.3,1X))
200   CONTINUE
      IF     (FINISH .GE. NCOL)  GO TO 300
      START   =  START  +  8
      FINISH  =  FINISH +  8
      GO TO 100
```

```
300   RETURN
      END
*
*************************************************
*  THE FOLLOWING ROUTINES HAVE BEEN TAKEN       *
*  FROM THE LINPACK MATHEMATICAL SOFTWARE.      *
*************************************************
*
      SUBROUTINE SSVDC(X,LDX,N,P,S,E,U,LDU,V,LDV,WORK,JOB,INFO)
      INTEGER LDX,N,P,LDU,LDV,JOB,INFO
      REAL X(LDX,1),S(1),E(1),U(LDU,1),V(LDV,1),WORK(1)
C
C
C     SSVDC IS A SUBROUTINE TO REDUCE A REAL NXP MATRIX X BY
C     ORTHOGONAL TRANSFORMATIONS U AND V TO DIAGONAL FORM.  THE
C     DIAGONAL ELEMENTS S(I) ARE THE SINGULAR VALUES OF X.  THE
C     COLUMNS OF U ARE THE CORRESPONDING LEFT SINGULAR VECTORS,
C     AND THE COLUMNS OF V THE RIGHT SINGULAR VECTORS.
C
C     ON ENTRY
C
C         X         REAL(LDX,P), WHERE LDX.GE.N.
C                   X CONTAINS THE MATRIX WHOSE SINGULAR VALUE
C                   DECOMPOSITION IS TO BE COMPUTED.  X IS
C                   DESTROYED BY SSVDC.
C
C         LDX       INTEGER.
C                   LDX IS THE LEADING DIMENSION OF THE ARRAY X.
C
C         N         INTEGER.
C                   N IS THE NUMBER OF COLUMNS OF THE MATRIX X.
C
C         P         INTEGER.
C                   P IS THE NUMBER OF ROWS OF THE MATRIX X.
C
C         LDU       INTEGER.
C                   LDU IS THE LEADING DIMENSION OF THE ARRAY U.
C                   (SEE BELOW).
C
C         LDV       INTEGER.
C                   LDV IS THE LEADING DIMENSION OF THE ARRAY V.
C                   (SEE BELOW).
C
C         WORK      REAL(N).
C                   WORK IS A SCRATCH ARRAY.
C
C         JOB       INTEGER.
C                   JOB CONTROLS THE COMPUTATION OF THE SINGULAR
C                   VECTORS.  IT HAS THE DECIMAL EXPANSION AB
C                   WITH THE FOLLOWING MEANING
C
C                        A.EQ.0     DO NOT COMPUTE THE LEFT SINGULAR
C                                   VECTORS.
```

```
C                         A.EQ.1    RETURN THE N LEFT SINGULAR VECTORS
C                                   IN U.
C                         A.GE.2    RETURN THE FIRST MIN(N,P) SINGULAR
C                                   VECTORS IN U.
C                         B.EQ.0    DO NOT COMPUTE THE RIGHT SINGULAR
C                                   VECTORS.
C                         B.EQ.1    RETURN THE RIGHT SINGULAR VECTORS
C                                   IN V.
C
C     ON RETURN
C
C          S          REAL(MM), WHERE MM=MIN(N+1,P).
C                     THE FIRST MIN(N,P) ENTRIES OF S CONTAIN THE
C                     SINGULAR VALUES OF X ARRANGED IN DESCENDING
C                     ORDER OF MAGNITUDE.
C
C          E          REAL(P).
C                     E ORDINARILY CONTAINS ZEROS.  HOWEVER SEE THE
C                     DISCUSSION OF INFO FOR EXCEPTIONS.
C
C          U          REAL(LDU,K), WHERE LDU.GE.N.  IF JOBA.EQ.1 THEN
C                                   K.EQ.N, IF JOBA.GE.2 THEN
C                                   K.EQ.MIN(N,P).
C                     U CONTAINS THE MATRIX OF RIGHT SINGULAR VECTORS.
C                     U IS NOT REFERENCED IF JOBA.EQ.0.  IF N.LE.P
C                     OR IF JOBA.EQ.2, THEN U MAY BE IDENTIFIED WITH X
C                     IN THE SUBROUTINE CALL.
C
C          V          REAL(LDV,P), WHERE LDV.GE.P.
C                     V CONTAINS THE MATRIX OF RIGHT SINGULAR VECTORS.
C                     V IS NOT REFERENCED IF JOB.EQ.0.  IF P.LE.N,
C                     THEN V MAY BE IDENTIFIED WITH X IN THE
C                     SUBROUTINE CALL.
C
C          INFO       INTEGER.
C                     THE SINGULAR VALUES (AND THEIR CORRESPONDING
C                     SINGULAR VECTORS) S(INFO+1),S(INFO+2),...,S(M)
C                     ARE CORRECT (HERE M=MIN(N,P)).  THUS IF
C                     INFO.EQ.0, ALL THE SINGULAR VALUES AND THEIR
C                     VECTORS ARE CORRECT.  IN ANY EVENT, THE MATRIX
C                     B = TRANS(U)*X*V IS THE BIDIAGONAL MATRIX
C                     WITH THE ELEMENTS OF S ON ITS DIAGONAL AND THE
C                     ELEMENTS OF E ON ITS SUPER-DIAGONAL (TRANS(U)
C                     IS THE TRANSPOSE OF U).  THUS THE SINGULAR
C                     VALUES OF X AND B ARE THE SAME.
C
C     LINPACK. THIS VERSION DATED 03/19/79 .
C     G.W. STEWART, UNIVERSITY OF MARYLAND, ARGONNE NATIONAL LAB.
C
C     ***** USES THE FOLLOWING FUNCTIONS AND SUBPROGRAMS.
C
C     EXTERNAL SROT
C     BLAS SAXPY,SDOT,SSCAL,SSWAP,SNRM2,SROTG
```

```
C     FORTRAN ABS,AMAX1,MAX0,MIN0,MOD,SQRT
C
C     INTERNAL VARIABLES
C
      INTEGER I,ITER,J,JOBU,K,KASE,KK,L,LL,LLS,LM1,LP1,LS,LU,M,MAXIT,
     *        MM,MM1,MP1,NCT,NCTP1,NCU,NRT,NRTP1
      REAL SDOT,T,R
      REAL B,C,CS,EL,EMM1,F,G,SNRM2,SCALE,SHIFT,SL,SM,SN,SMM1,T1,TEST,
     *     ZTEST
      LOGICAL WANTU,WANTV
C
C
C     SET THE MAXIMUM NUMBER OF ITERATIONS.
C
      MAXIT = 30
C
C     DETERMINE WHAT IS TO BE COMPUTED.
C
      WANTU = .FALSE.
      WANTV = .FALSE.
      JOBU = MOD(JOB,100)/10
      NCU = N
      IF (JOBU .GT. 1) NCU = MIN0(N,P)
      IF (JOBU .NE. 0) WANTU = .TRUE.
      IF (MOD(JOB,10) .NE. 0) WANTV = .TRUE.
C
C     REDUCE X TO BIDIAGONAL FORM, STORING THE DIAGONAL ELEMENTS
C     IN S AND THE SUPER-DIAGONAL ELEMENTS IN E.
C
      INFO = 0
      NCT = MIN0(N-1,P)
      NRT = MAX0(0,MIN0(P-2,N))
      LU = MAX0(NCT,NRT)
      IF (LU .LT. 1) GO TO 170
      DO 160 L = 1, LU
         LP1 = L + 1
         IF (L .GT. NCT) GO TO 20
C
C           COMPUTE THE TRANSFORMATION FOR THE L-TH COLUMN AND
C           PLACE THE L-TH DIAGONAL IN S(L).
C
            S(L) = SNRM2(N-L+1,X(L,L),1)
            IF (S(L) .EQ. 0.0E0) GO TO 10
               IF (X(L,L) .NE. 0.0E0) S(L) = SIGN(S(L),X(L,L))
               CALL SSCAL(N-L+1,1.0E0/S(L),X(L,L),1)
               X(L,L) = 1.0E0 + X(L,L)
   10       CONTINUE
            S(L) = -S(L)
   20    CONTINUE
         IF (P .LT. LP1) GO TO 50
         DO 40 J = LP1, P
            IF (L .GT. NCT) GO TO 30
            IF (S(L) .EQ. 0.0E0) GO TO 30
```

```
C
C              APPLY THE TRANSFORMATION.
C
               T = -SDOT(N-L+1,X(L,L),1,X(L,J),1)/X(L,L)
               CALL SAXPY(N-L+1,T,X(L,L),1,X(L,J),1)
   30       CONTINUE
C
C           PLACE THE L-TH ROW OF X INTO  E FOR THE
C           SUBSEQUENT CALCULATION OF THE ROW TRANSFORMATION.
C
            E(J) = X(L,J)
   40    CONTINUE
   50    CONTINUE
         IF (.NOT.WANTU .OR. L .GT. NCT) GO TO 70
C
C           PLACE THE TRANSFORMATION IN U FOR SUBSEQUENT BACK
C           MULTIPLICATION.
C
            DO 60 I = L, N
               U(I,L) = X(I,L)
   60       CONTINUE
   70    CONTINUE
         IF (L .GT. NRT) GO TO 150
C
C           COMPUTE THE L-TH ROW TRANSFORMATION AND PLACE THE
C           L-TH SUPER-DIAGONAL IN E(L).
C
            E(L) = SNRM2(P-L,E(LP1),1)
            IF (E(L) .EQ. 0.0E0) GO TO 80
               IF (E(LP1) .NE. 0.0E0) E(L) = SIGN(E(L),E(LP1))
               CALL SSCAL(P-L,1.0E0/E(L),E(LP1),1)
               E(LP1) = 1.0E0 + E(LP1)
   80       CONTINUE
            E(L) = -E(L)
            IF (LP1 .GT. N .OR. E(L) .EQ. 0.0E0) GO TO 120
C
C              APPLY THE TRANSFORMATION.
C
               DO 90 I = LP1, N
                  WORK(I) = 0.0E0
   90          CONTINUE
               DO 100 J = LP1, P
                  CALL SAXPY(N-L,E(J),X(LP1,J),1,WORK(LP1),1)
  100          CONTINUE
               DO 110 J = LP1, P
                  CALL SAXPY(N-L,-E(J)/E(LP1),WORK(LP1),1,X(LP1,J),1)
  110          CONTINUE
  120       CONTINUE
            IF (.NOT.WANTV) GO TO 140
C
C              PLACE THE TRANSFORMATION IN V FOR SUBSEQUENT
C              BACK MULTIPLICATION.
C
```

```
                  DO 130 I = LP1, P
                     V(I,L) = E(I)
  130             CONTINUE
  140          CONTINUE
  150       CONTINUE
  160    CONTINUE
  170 CONTINUE
C
C     SET UP THE FINAL BIDIAGONAL MATRIX OR ORDER M.
C
      M = MIN0(P,N+1)
      NCTP1 = NCT + 1
      NRTP1 = NRT + 1
      IF (NCT .LT. P) S(NCTP1) = X(NCTP1,NCTP1)
      IF (N .LT. M) S(M) = 0.0E0
      IF (NRTP1 .LT. M) E(NRTP1) = X(NRTP1,M)
      E(M) = 0.0E0
C
C     IF REQUIRED, GENERATE U.
C
      IF (.NOT.WANTU) GO TO 300
         IF (NCU .LT. NCTP1) GO TO 200
         DO 190 J = NCTP1, NCU
            DO 180 I = 1, N
               U(I,J) = 0.0E0
  180       CONTINUE
            U(J,J) = 1.0E0
  190    CONTINUE
  200    CONTINUE
         IF (NCT .LT. 1) GO TO 290
         DO 280 LL = 1, NCT
            L = NCT - LL + 1
            IF (S(L) .EQ. 0.0E0) GO TO 250
               LP1 = L + 1
               IF (NCU .LT. LP1) GO TO 220
               DO 210 J = LP1, NCU
                  T = -SDOT(N-L+1,U(L,L),1,U(L,J),1)/U(L,L)
                  CALL SAXPY(N-L+1,T,U(L,L),1,U(L,J),1)
  210          CONTINUE
  220          CONTINUE
               CALL SSCAL(N-L+1,-1.0E0,U(L,L),1)
               U(L,L) = 1.0E0 + U(L,L)
               LM1 = L - 1
               IF (LM1 .LT. 1) GO TO 240
               DO 230 I = 1, LM1
                  U(I,L) = 0.0E0
  230          CONTINUE
  240          CONTINUE
            GO TO 270
  250       CONTINUE
               DO 260 I = 1, N
                  U(I,L) = 0.0E0
  260          CONTINUE
```

```
                  U(L,L) = 1.0E0
  270          CONTINUE
  280       CONTINUE
  290       CONTINUE
  300    CONTINUE
C
C        IF IT IS REQUIRED, GENERATE V.
C
         IF (.NOT.WANTV) GO TO 350
            DO 340 LL = 1, P
               L = P - LL + 1
               LP1 = L + 1
               IF (L .GT. NRT) GO TO 320
               IF (E(L) .EQ. 0.0E0) GO TO 320
                  DO 310 J = LP1, P
                     T = -SDOT(P-L,V(LP1,L),1,V(LP1,J),1)/V(LP1,L)
                     CALL SAXPY(P-L,T,V(LP1,L),1,V(LP1,J),1)
  310             CONTINUE
  320          CONTINUE
               DO 330 I = 1, P
                  V(I,L) = 0.0E0
  330          CONTINUE
               V(L,L) = 1.0E0
  340       CONTINUE
  350    CONTINUE
C
C        MAIN ITERATION LOOP FOR THE SINGULAR VALUES.
C
         MM = M
         ITER = 0
  360    CONTINUE
C
C           QUIT IF ALL THE SINGULAR VALUES HAVE BEEN FOUND.
C
C        ...EXIT
            IF (M .EQ. 0) GO TO 620
C
C           IF TOO MANY ITERATIONS HAVE BEEN PERFORMED, SET
C           FLAG AND RETURN.
C
            IF (ITER .LT. MAXIT) GO TO 370
               INFO = M
C        ......EXIT
               GO TO 620
  370       CONTINUE
C
C           THIS SECTION OF THE PROGRAM INSPECTS FOR
C           NEGLIGIBLE ELEMENTS IN THE S AND E ARRAYS.  ON
C           COMPLETION THE VARIABLES KASE AND L ARE SET AS FOLLOWS.
C
C              KASE = 1     IF S(M) AND E(L-1) ARE NEGLIGIBLE AND L.LT.M
C              KASE = 2     IF S(L) IS NEGLIGIBLE AND L.LT.M
C              KASE = 3     IF E(L-1) IS NEGLIGIBLE, L.LT.M, AND
```

```
C                        S(L), ..., S(M) ARE NOT NEGLIGIBLE (QR STEP).
C           KASE = 4     IF E(M-1) IS NEGLIGIBLE (CONVERGENCE).
C
         DO 390 LL = 1, M
            L = M - LL
C        ...EXIT
            IF (L .EQ. 0) GO TO 400
            TEST = ABS(S(L)) + ABS(S(L+1))
            ZTEST = TEST + ABS(E(L))
            IF (ZTEST .NE. TEST) GO TO 380
               E(L) = 0.0E0
C        ......EXIT
               GO TO 400
  380       CONTINUE
  390    CONTINUE
  400    CONTINUE
         IF (L .NE. M - 1) GO TO 410
            KASE = 4
         GO TO 480
  410    CONTINUE
            LP1 = L + 1
            MP1 = M + 1
            DO 430 LLS = LP1, MP1
               LS = M - LLS + LP1
C           ...EXIT
               IF (LS .EQ. L) GO TO 440
               TEST = 0.0E0
               IF (LS .NE. M) TEST = TEST + ABS(E(LS))
               IF (LS .NE. L + 1) TEST = TEST + ABS(E(LS-1))
               ZTEST = TEST + ABS(S(LS))
               IF (ZTEST .NE. TEST) GO TO 420
                  S(LS) = 0.0E0
C           ......EXIT
                  GO TO 440
  420          CONTINUE
  430       CONTINUE
  440       CONTINUE
            IF (LS .NE. L) GO TO 450
               KASE = 3
            GO TO 470
  450       CONTINUE
            IF (LS .NE. M) GO TO 460
               KASE = 1
            GO TO 470
  460       CONTINUE
               KASE = 2
               L = LS
  470       CONTINUE
  480    CONTINUE
         L = L + 1
C
C        PERFORM THE TASK INDICATED BY KASE.
C
```

```
      GO TO (490,520,540,570), KASE
C
C     DEFLATE NEGLIGIBLE S(M).
C
  490 CONTINUE
         MM1 = M - 1
         F = E(M-1)
         E(M-1) = 0.0E0
         DO 510 KK = L, MM1
            K = MM1 - KK + L
            T1 = S(K)
            CALL SROTG(T1,F,CS,SN)
            S(K) = T1
            IF (K .EQ. L) GO TO 500
               F = -SN*E(K-1)
               E(K-1) = CS*E(K-1)
  500       CONTINUE
            IF (WANTV) CALL SROT(P,V(1,K),1,V(1,M),1,CS,SN)
  510    CONTINUE
      GO TO 610
C
C     SPLIT AT NEGLIGIBLE S(L).
C
  520 CONTINUE
         F = E(L-1)
         E(L-1) = 0.0E0
         DO 530 K = L, M
            T1 = S(K)
            CALL SROTG(T1,F,CS,SN)
            S(K) = T1
            F = -SN*E(K)
            E(K) = CS*E(K)
            IF (WANTU) CALL SROT(N,U(1,K),1,U(1,L-1),1,CS,SN)
  530    CONTINUE
      GO TO 610
C
C     PERFORM ONE QR STEP.
C
  540 CONTINUE
C
C        CALCULATE THE SHIFT.
C
         SCALE = AMAX1(ABS(S(M)),ABS(S(M-1)),ABS(E(M-1)),ABS(S(L)),
     *                 ABS(E(L)))
         SM = S(M)/SCALE
         SMM1 = S(M-1)/SCALE
         EMM1 = E(M-1)/SCALE
         SL = S(L)/SCALE
         EL = E(L)/SCALE
         B = ((SMM1 + SM)*(SMM1 - SM) + EMM1**2)/2.0E0
         C = (SM*EMM1)**2
         SHIFT = 0.0E0
         IF (B .EQ. 0.0E0 .AND. C .EQ. 0.0E0) GO TO 550
```

```
               SHIFT = SQRT(B**2+C)
               IF (B .LT. 0.0E0) SHIFT = -SHIFT
               SHIFT = C/(B + SHIFT)
  550       CONTINUE
            F = (SL + SM)*(SL - SM) - SHIFT
            G = SL*EL
C
C           CHASE ZEROS.
C
            MM1 = M - 1
            DO 560 K = L, MM1
               CALL SROTG(F,G,CS,SN)
               IF (K .NE. L) E(K-1) = F
               F = CS*S(K) + SN*E(K)
               E(K) = CS*E(K) - SN*S(K)
               G = SN*S(K+1)
               S(K+1) = CS*S(K+1)
               IF (WANTV) CALL SROT(P,V(1,K),1,V(1,K+1),1,CS,SN)
               CALL SROTG(F,G,CS,SN)
               S(K) = F
               F = CS*E(K) + SN*S(K+1)
               S(K+1) = -SN*E(K) + CS*S(K+1)
               G = SN*E(K+1)
               E(K+1) = CS*E(K+1)
               IF (WANTU .AND. K .LT. N)
     *            CALL SROT(N,U(1,K),1,U(1,K+1),1,CS,SN)
  560       CONTINUE
            E(M-1) = F
            ITER = ITER + 1
         GO TO 610
C
C        CONVERGENCE.
C
  570    CONTINUE
C
C           MAKE THE SINGULAR VALUE  POSITIVE.
C
            IF (S(L) .GE. 0.0E0) GO TO 580
               S(L) = -S(L)
               IF (WANTV) CALL SSCAL(P,-1.0E0,V(1,L),1)
  580       CONTINUE
C
C           ORDER THE SINGULAR VALUE.
C
  590       IF (L .EQ. MM) GO TO 600
C           ...EXIT
               IF (S(L) .GE. S(L+1)) GO TO 600
               T = S(L)
               S(L) = S(L+1)
               S(L+1) = T
               IF (WANTV .AND. L .LT. P)
     *            CALL SSWAP(P,V(1,L),1,V(1,L+1),1)
               IF (WANTU .AND. L .LT. N)
```

```
     *            CALL SSWAP(N,U(1,L),1,U(1,L+1),1)
                  L = L + 1
               GO TO 590
  600          CONTINUE
               ITER = 0
               M = M - 1
  610       CONTINUE
        GO TO 360
  620   CONTINUE
        RETURN
        END
        REAL FUNCTION SNRM2 ( N, SX, INCX)
        INTEGER          NEXT,      N ,     INCX
        INTEGER          NN  ,      I ,      J
        REAL    SX(1),  CUTLO, CUTHI, HITEST, SUM, XMAX, ZERO, ONE
        DATA    ZERO, ONE /0.0E0, 1.0E0/
C
C       EUCLIDEAN NORM OF THE N-VECTOR STORED IN SX() WITH STORAGE
C       INCREMENT INCX .
C       IF    N .LE. 0 RETURN WITH RESULT = 0.
C       IF N .GE. 1 THEN INCX MUST BE .GE. 1
C
C            C.L.LAWSON, 1978 JAN 08
C
C       FOUR PHASE METHOD     USING TWO BUILT-IN CONSTANTS THAT ARE
C       HOPEFULLY APPLICABLE TO ALL MACHINES.
C           CUTLO = MAXIMUM OF  SQRT(U/EPS)  OVER ALL KNOWN MACHINES.
C           CUTHI = MINIMUM OF  SQRT(V)      OVER ALL KNOWN MACHINES.
C       WHERE
C           EPS = SMALLEST NO. SUCH THAT EPS + 1. .GT. 1.
C           U   = SMALLEST POSITIVE NO.   (UNDERFLOW LIMIT)
C           V   = LARGEST  NO.            (OVERFLOW  LIMIT)
C
C       BRIEF OUTLINE OF ALGORITHM..
C
C       PHASE 1    SCANS ZERO COMPONENTS.
C       MOVE TO PHASE 2 WHEN A COMPONENT IS NONZERO AND .LE. CUTLO
C       MOVE TO PHASE 3 WHEN A COMPONENT IS .GT. CUTLO
C       MOVE TO PHASE 4 WHEN A COMPONENT IS .GE. CUTHI/M
C       WHERE M = N FOR X() REAL AND M = 2*N FOR COMPLEX.
C
C       VALUES FOR CUTLO AND CUTHI..
C       FROM THE ENVIRONMENTAL PARAMETERS LISTED IN THE IMSL CONVERTER
C       DOCUMENT THE LIMITING VALUES ARE AS FOLLOWS..
C       CUTLO, S.P.   U/EPS = 2**(-102) FOR  HONEYWELL.  CLOSE SECONDS ARE
C                     UNIVAC AND DEC AT 2**(-103)
C                     THUS CUTLO = 2**(-51) = 4.44089E-16
C       CUTHI, S.P.   V = 2**127 FOR UNIVAC, HONEYWELL, AND DEC.
C                     THUS CUTHI = 2**(63.5) = 1.30438E19
C       CUTLO, D.P.   U/EPS = 2**(-67) FOR HONEYWELL AND DEC.
C                     THUS CUTLO = 2**(-33.5) = 8.23181D-11
C       CUTHI, D.P.   SAME AS S.P.  CUTHI = 1.30438D19
C       DATA CUTLO, CUTHI / 8.232D-11,  1.304D19 /
```

```
C     DATA CUTLO, CUTHI / 4.441E-16,  1.304E19 /
      DATA CUTLO, CUTHI / 4.441E-16,  1.304E19 /
C
      IF(N .GT. 0) GO TO 10
         SNRM2  = ZERO
         GO TO 300
C
   10 ASSIGN 30 TO NEXT
      SUM = ZERO
      NN = N * INCX
C                                                 BEGIN MAIN LOOP
      I = 1
   20    GO TO NEXT,(30, 50, 70, 110)
   30 IF( ABS(SX(I)) .GT. CUTLO) GO TO 85
      ASSIGN 50 TO NEXT
      XMAX = ZERO
C
C                       PHASE 1.  SUM IS ZERO
C
   50 IF( SX(I) .EQ. ZERO) GO TO 200
      IF( ABS(SX(I)) .GT. CUTLO) GO TO 85
C
C                                PREPARE FOR PHASE 2.
      ASSIGN 70 TO NEXT
      GO TO 105
C
C                                PREPARE FOR PHASE 4.
C
  100 I = J
      ASSIGN 110 TO NEXT
      SUM = (SUM / SX(I)) / SX(I)
  105 XMAX = ABS(SX(I))
      GO TO 115
C
C                  PHASE 2.  SUM IS SMALL.
C                            SCALE TO AVOID DESTRUCTIVE UNDERFLOW.
C
   70 IF( ABS(SX(I)) .GT. CUTLO ) GO TO 75
C
C                    COMMON CODE FOR PHASES 2 AND 4.
C                    IN PHASE 4 SUM IS LARGE.  SCALE TO AVOID OVERFLOW.
C
  110 IF( ABS(SX(I)) .LE. XMAX ) GO TO 115
         SUM = ONE + SUM * (XMAX / SX(I))**2
         XMAX = ABS(SX(I))
         GO TO 200
C
  115 SUM = SUM + (SX(I)/XMAX)**2
      GO TO 200
C
C
C                  PREPARE FOR PHASE 3.
C
```

```
   75 SUM = (SUM * XMAX) * XMAX
C
C
C     FOR REAL OR D.P. SET HITEST = CUTHI/N
C     FOR COMPLEX      SET HITEST = CUTHI/(2*N)
C
   85 HITEST = CUTHI/FLOAT( N )
C
C                     PHASE 3.  SUM IS MID-RANGE.  NO SCALING.
C
      DO 95 J =I,NN,INCX
      IF(ABS(SX(J)) .GE. HITEST) GO TO 100
   95    SUM = SUM + SX(J)**2
      SNRM2 = SQRT( SUM )
      GO TO 300
C
  200 CONTINUE
      I = I + INCX
      IF ( I .LE. NN ) GO TO 20
C
C               END OF MAIN LOOP.
C
C               COMPUTE SQUARE ROOT AND ADJUST FOR SCALING.
C
      SNRM2 = XMAX * SQRT(SUM)
  300 CONTINUE
      RETURN
      END
      SUBROUTINE  SSCAL(N,SA,SX,INCX)
C
C     SCALES A VECTOR BY A CONSTANT.
C     USES UNROLLED LOOPS FOR INCREMENT EQUAL TO 1.
C     JACK DONGARRA, LINPACK, 3/11/78.
C
      REAL SA,SX(1)
      INTEGER I,INCX,M,MP1,N,NINCX
C
      IF(N.LE.0)RETURN
      IF(INCX.EQ.1)GO TO 20
C
C        CODE FOR INCREMENT NOT EQUAL TO 1
C
      NINCX = N*INCX
      DO 10 I = 1,NINCX,INCX
        SX(I) = SA*SX(I)
   10 CONTINUE
      RETURN
C
C        CODE FOR INCREMENT EQUAL TO 1
C
C
C        CLEAN-UP LOOP
C
```

```
   20 M = MOD(N,5)
      IF( M .EQ. 0 ) GO TO 40
      DO 30 I = 1,M
        SX(I) = SA*SX(I)
   30 CONTINUE
      IF( N .LT. 5 ) RETURN
   40 MP1 = M + 1
      DO 50 I = MP1,N,5
        SX(I) = SA*SX(I)
        SX(I + 1) = SA*SX(I + 1)
        SX(I + 2) = SA*SX(I + 2)
        SX(I + 3) = SA*SX(I + 3)
        SX(I + 4) = SA*SX(I + 4)
   50 CONTINUE
      RETURN
      END
      REAL FUNCTION SDOT(N,SX,INCX,SY,INCY)
C
C     FORMS THE DOT PRODUCT OF TWO VECTORS.
C     USES UNROLLED LOOPS FOR INCREMENTS EQUAL TO ONE.
C     JACK DONGARRA, LINPACK, 3/11/78.
C
      REAL SX(1),SY(1),STEMP
      INTEGER I,INCX,INCY,IX,IY,M,MP1,N
C
      STEMP = 0.0E0
      SDOT = 0.0E0
      IF(N.LE.0)RETURN
      IF(INCX.EQ.1.AND.INCY.EQ.1)GO TO 20
C
C        CODE FOR UNEQUAL INCREMENTS OR EQUAL INCREMENTS
C          NOT EQUAL TO 1
C
      IX = 1
      IY = 1
      IF(INCX.LT.0)IX = (-N+1)*INCX + 1
      IF(INCY.LT.0)IY = (-N+1)*INCY + 1
      DO 10 I = 1,N
        STEMP = STEMP + SX(IX)*SY(IY)
        IX = IX + INCX
        IY = IY + INCY
   10 CONTINUE
      SDOT = STEMP
      RETURN
C
C        CODE FOR BOTH INCREMENTS EQUAL TO 1
C
C
C        CLEAN-UP LOOP
C
   20 M = MOD(N,5)
      IF( M .EQ. 0 ) GO TO 40
      DO 30 I = 1,M
```

```
        STEMP = STEMP + SX(I)*SY(I)
   30 CONTINUE
      IF( N .LT. 5 ) GO TO 60
   40 MP1 = M + 1
      DO 50 I = MP1,N,5
        STEMP = STEMP + SX(I)*SY(I) + SX(I + 1)*SY(I + 1) +
     *   SX(I + 2)*SY(I + 2) + SX(I + 3)*SY(I + 3) + SX(I + 4)*SY(I + 4)
   50 CONTINUE
   60 SDOT = STEMP
      RETURN
      END
      SUBROUTINE SAXPY(N,SA,SX,INCX,SY,INCY)
C
C     CONSTANT TIMES A VECTOR PLUS A VECTOR.
C     USES UNROLLED LOOP FOR INCREMENTS EQUAL TO ONE.
C     JACK DONGARRA, LINPACK, 3/11/78.
C
      REAL SX(1),SY(1),SA
      INTEGER I,INCX,INCY,IX,IY,M,MP1,N
C
      IF(N.LE.0)RETURN
      IF (SA .EQ. 0.0) RETURN
      IF(INCX.EQ.1.AND.INCY.EQ.1)GO TO 20
C
C        CODE FOR UNEQUAL INCREMENTS OR EQUAL INCREMENTS
C          NOT EQUAL TO 1
C
      IX = 1
      IY = 1
      IF(INCX.LT.0)IX = (-N+1)*INCX + 1
      IF(INCY.LT.0)IY = (-N+1)*INCY + 1
      DO 10 I = 1,N
        SY(IY) = SY(IY) + SA*SX(IX)
        IX = IX + INCX
        IY = IY + INCY
   10 CONTINUE
      RETURN
C
C        CODE FOR BOTH INCREMENTS EQUAL TO 1
C
C
C        CLEAN-UP LOOP
C
   20 M = MOD(N,4)
      IF( M .EQ. 0 ) GO TO 40
      DO 30 I = 1,M
        SY(I) = SY(I) + SA*SX(I)
   30 CONTINUE
      IF( N .LT. 4 ) RETURN
   40 MP1 = M + 1
      DO 50 I = MP1,N,4
        SY(I) = SY(I) + SA*SX(I)
        SY(I + 1) = SY(I + 1) + SA*SX(I + 1)
```

```
          SY(I + 2) = SY(I + 2) + SA*SX(I + 2)
          SY(I + 3) = SY(I + 3) + SA*SX(I + 3)
   50 CONTINUE
      RETURN
      END
      SUBROUTINE SROTG(SA,SB,C,S)
C
C     CONSTRUCT GIVENS PLANE ROTATION.
C     JACK DONGARRA, LINPACK, 3/11/78.
C
      REAL SA,SB,C,S,ROE,SCALE,R,Z
C
      ROE = SB
      IF( ABS(SA) .GT. ABS(SB) ) ROE = SA
      SCALE = ABS(SA) + ABS(SB)
      IF( SCALE .NE. 0.0 ) GO TO 10
         C = 1.0
         S = 0.0
         R = 0.0
         GO TO 20
   10 R = SCALE*SQRT((SA/SCALE)**2 + (SB/SCALE)**2)
      R = SIGN(1.0,ROE)*R
      C = SA/R
      S = SB/R
   20 Z = 1.0
      IF( ABS(SA) .GT. ABS(SB) ) Z = S
      IF( ABS(SB) .GE. ABS(SA) .AND. C .NE. 0.0 ) Z = 1.0/C
      SA = R
      SB = Z
      RETURN
      END
      SUBROUTINE  SROT (N,SX,INCX,SY,INCY,C,S)
C
C     APPLIES A PLANE ROTATION.
C     JACK DONGARRA, LINPACK, 3/11/78.
C
      REAL SX(1),SY(1),STEMP,C,S
      INTEGER I,INCX,INCY,IX,IY,N
C
      IF(N.LE.0)RETURN
      IF(INCX.EQ.1.AND.INCY.EQ.1)GO TO 20
C
C       CODE FOR UNEQUAL INCREMENTS OR EQUAL INCREMENTS NOT EQUAL
C         TO 1
C
      IX = 1
      IY = 1
      IF(INCX.LT.0)IX = (-N+1)*INCX + 1
      IF(INCY.LT.0)IY = (-N+1)*INCY + 1
      DO 10 I = 1,N
        STEMP = C*SX(IX) + S*SY(IY)
        SY(IY) = C*SY(IY) - S*SX(IX)
        SX(IX) = STEMP
```

```
        IX = IX + INCX
        IY = IY + INCY
   10 CONTINUE
      RETURN
C
C       CODE FOR BOTH INCREMENTS EQUAL TO 1
C
   20 DO 30 I = 1,N
        STEMP = C*SX(I) + S*SY(I)
        SY(I) = C*SY(I) - S*SX(I)
        SX(I) = STEMP
   30 CONTINUE
      RETURN
      END
      SUBROUTINE  SSWAP (N,SX,INCX,SY,INCY)
C
C     INTERCHANGES TWO VECTORS.
C     USES UNROLLED LOOPS FOR INCREMENTS EQUAL TO 1.
C     JACK DONGARRA, LINPACK, 3/11/78.
C
      REAL SX(1),SY(1),STEMP
      INTEGER I,INCX,INCY,IX,IY,M,MP1,N
C
      IF(N.LE.0)RETURN
      IF(INCX.EQ.1.AND.INCY.EQ.1)GO TO 20
C
C       CODE FOR UNEQUAL INCREMENTS OR EQUAL INCREMENTS NOT EQUAL
C         TO 1
C
      IX = 1
      IY = 1
      IF(INCX.LT.0)IX = (-N+1)*INCX + 1
      IF(INCY.LT.0)IY = (-N+1)*INCY + 1
      DO 10 I = 1,N
        STEMP = SX(IX)
        SX(IX) = SY(IY)
        SY(IY) = STEMP
        IX = IX + INCX
        IY = IY + INCY
   10 CONTINUE
      RETURN
C
C       CODE FOR BOTH INCREMENTS EQUAL TO 1
C
C
C       CLEAN-UP LOOP
C
   20 M = MOD(N,3)
      IF( M .EQ. 0 ) GO TO 40
      DO 30 I = 1,M
        STEMP = SX(I)
        SX(I) = SY(I)
        SY(I) = STEMP
```

```
30 CONTINUE
   IF( N .LT. 3 ) RETURN
40 MP1 = M + 1
   DO 50 I = MP1,N,3
     STEMP = SX(I)
     SX(I) = SY(I)
     SY(I) = STEMP
     STEMP = SX(I + 1)
     SX(I + 1) = SY(I + 1)
     SY(I + 1) = STEMP
     STEMP = SX(I + 2)
     SX(I + 2) = SY(I + 2)
     SY(I + 2) = STEMP
50 CONTINUE
   RETURN
   END
```

REFERENCES

[1] E.C.Y. Tse, J. Medanic, and W.R. Perkins, "Generalized Hessenberg Transformations for Reduced Order Modeling of Large Scale Systems," Int. J. Control, Vol. 27, 1978, pp. 493-512.

[2] D. Lindner, W.R. Perkins, and J. Medanic, "Chained Aggregation and Three-Control-Component Design: A Geometric Analysis," Int. J. Control, Vol. 35, 1982, pp. 621-636.

[3] E. Tse, W.R. Perkins, J. Medanic, and D. Lindner, "Hierarchical Control of Large Scale Systems By Chained Aggregation," IFAC Symposium on Large Scale Systems: Theory and Applications, Toulouse, France, 1980, pp. 203-210.

[4] D.K. Lindner and W.R. Perkins, "System Structural Decomposition By Chained Aggregation," 1982 IEEE Int. Large Scale Systems Symposium, Virginia Beach, VA.

[5] D. Lindner, J. Medanic, and W.R. Perkins, "Decomposition of a Class of Nonlinear Systems via Chained Aggregation," 3rd IFAC/IFORS Symposium on Large Scale Systems, Warsaw, Poland, 1983.

[6] D.K. Lindner, "Chained Aggregation and Control System Design: A Geometric Approach," Report R-966 (DC-56), Decision and Control Laboratory, University of Illinois, Urbana, Illinois, 1982.

[7] D.K. Lindner, J. Medanic, and W.R. Perkins, "Three Control Component Design for Interconnected Systems," Submitted for publication.

[8] V.C. Klema and A.J. Laub, "The Singular Value Decomposition: Its Computation and Some Applications," IEEE Trans. Automatic Control, Vol. AC-25, 1980, pp. 164-176.

[9] G. Strang, Linear Algebra and Its Applications, New York: Academic Press, 1980.

[10] P.M. Van Dooren, "The Generalized Eigenstructure Problem in Linear System Theory," IEEE Trans. Automatic Control, Vol. AC-26, 1981, pp. 111-129.

[11] G.W. Stewart, Introduction to Matrix Computations. New York: Academic Press, 1973.

[12] C.C. Paige, "Properties of Numerical Algorithms Related to Computing Controllability," IEEE Trans. Automatic Control, Vol. AC-26, 1981, pp. 130-138.

[13] W.M. Wonham, Linear Multivariable Theory A Geometric Approach, 2nd ed., New York: Springer, 1979.

[14] P.M. Van Dooren, A. Emami-Naeini, and L. Silverman, "Stable Extraction of the Kronecker Structure of Pencils," in Proc. 17th IEEE Conf. on Decision and Control, January 1979, pp. 521-524.

[15] M.J. Balas, "Trends in Large Space Structure Control Theory: Fondest Hopes, Wildest Dreams," IEEE Trans. Automatic Control, Vol. AC-27, 1982, pp. 522-535.

[16] M. Balas and S. Gunter, "Attitude Stabilization of Large Flexible Spacecraft," J. Guidance Control, vol. 4, 1981, pp. 561-564.

[17] M.J. Balas, "Feedback Control of Flexible Systems," IEEE Trans. Automatic Control, Vol. AC-23, 1978, pp. 673-679.

[18] R. Gran and M. Rossi, "A Survey of the Large Structures Control Problem," in Proc. 18th IEEE Conf. on Decision and Control, Dec. 1979, pp. 1002-1007.

[19] R.J. Benhabib, "Discrete Large Space Structure Control System Design Using Positivity," in Proc. 20th IEEE Conf. on Decision and Control, Dec. 1981, pp. 715-724.

[20] S.M. Joshi and N.J. Groom, "Optimal Member Damper Controller Design for Large Space Structures," J. Guidance Contr., vol. 3, 1980, pp. 378-380.

[21] P. Likins and H. Bouvier, "Attitude Control of Nonrigid Spacecraft," AIAA Aeronaut. Astronaut., vol. 9, No. 5, 1971, pp. 64-71.

[22] S.M. Seltzer, "Dynamics and Control of Large Space Structures: An Overview," Journal Astronaut. Sciences, vol. 27, No. 2, 1979, pp. 95-101.

[23] R.J. Herzberg, K.F. Johansen, and R.C. Stroud, "Dynamics and Control of Large Satellites," AIAA Aeronaut. Astronaut., vol. 16, No. 10, 1978, pp. 35-39.

[24] E.J. Davison, "A Method for Simplifying Linear Dynamic Systems," IEEE Trans. Automatic Control, Vol. AC-11, 1966, pp. 93-101.

[25] E.C.Y. Tse, "Model Reduction and Decentralized Control of Large Scale Systems Using Chained Aggregation," Report R-820, Decision and Control Laboratory, University of Illinois, Urbana, Illinois, 1978.

[26] G.W. Stewart, "Error and Perturbation Bounds for Subspaces Associated with Certain Eigenvalue Problems," SIAM Review, Vol. 15, 1973, pp. 727-764.

[27] W.E. Hopkins, Jr., "Output Feedback Pole-Placement in the Design of Compensators for Suboptimal Linear Quadratic Regulators," Report R-847, Decision and Control Laboratory, University of Illinois, Urbana, Illinois, 1979.

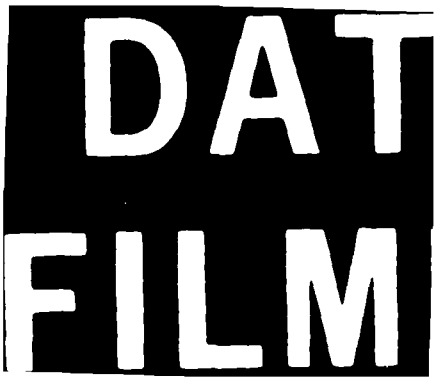
[28] W.E. Hopkins, Jr., J. Medanic, and W.R. Perkins, "Output Feedback Pole Placement in the Design of Suboptimal Linear Quadratic Regulators," Int. J. Control, Vol. 34, 1981, pp. 593-612.

[29] J. Medanic, "Design of Low Order Optimal Dynamic Regulators for Linear Time-Invariant Systems," Conf. on Information Theory and Systems, Johns Hopkins University, 1979, pp. 97-102.

[30] J. Medanic, "On Stabilization And Optimization By Output Feedback," Twelfth Asilomar Conf. on Circuits, Systems and Computers, 1978, pp. 412-416.

[31] J.J. Dongarra, J.R. Bunch, C.B. Moler, and G.W. Stewart, LINPACK Users' Guide, SIAM, Philadelphia, 1979.

[32] S.P. Bingulac and N. Gluhajic, "Computer Aided Design of Control System On Mini Computer Using the L-A-S Language," presented at IFAC Symposium on Computer Aided Design of Multivariable Technological Systems, Purdue University, Indiana, Sept. 15-17, 1982.


DAT
FILM